175 PTAS



CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

10090
9 788485 822836

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VIII-Fascículo 90

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones
	Francisco Martín
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 -

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 168510
Impreso en España-Printed in Spain-Octubre 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.





Sistemas expertos

Continuando con nuestra serie, centraremos nuestra atención en los programas denominados "sistemas expertos"

Un experto (un consejero hospitalario, un geólogo o un analista químico, p. ej.) suele ser una persona respetada. Los expertos dedican mucho tiempo a estudiar y practicar la disciplina elegida para llegar a ejercer eficientemente su trabajo. La utilización de expertos humanos presenta, no obstante, algunos problemas por su escaso número, su falibilidad, el pago de sus remuneraciones y, por supuesto, su calidad de mortales, que implica, a la larga, la pérdida definitiva de la destreza adquirida. Por este motivo es comprensible que a muchas empresas les atraiga la idea de codificar la pericia en programas de ordenador, para beneficiarse de poder trabajar con un gran cuerpo de conocimientos sin los inconvenientes propios de los expertos humanos.

El concepto de *sistema experto* nació en los años setenta, cuando los investigadores en el campo de la inteligencia artificial abandonaron, o pospusieron, la creación de máquinas inteligentes a nivel general y volcaron su interés en la solución de problemas del mundo real centrados en aspectos muy concretos. Por consiguiente, el sistema experto es uno de los primeros ejemplos de AI aplicada, y las técnicas para sistemas expertos se han extendido mucho más allá de los confines de los laboratorios de investigación en los que fueron concebidas. De hecho, en cierto modo los sistemas expertos han llevado a la AI al uso práctico cotidiano. Existen ya sistemas que superan a los seres humanos capacitados en diagnóstico médico, interpretación de espectrogramas de masa, clasificación de enfermedades de cultivos y otras muchas cosas más. Es interesante preguntarse, en consecuencia, cómo funcionan.

Típicamente, un sistema experto se basa en un amplio cuerpo de conocimientos sobre un área problemática específica. En general, este conocimiento se organiza como un conjunto de reglas que permitan que el sistema extraiga conclusiones a partir de datos o premisas dadas, capacitándolo, en consecuencia, para ofrecer un consejo inteligente o tomar decisiones inteligentes. Este enfoque al diseño de sistemas basado en el conocimiento representa un cambio evolutivo en la ciencia de los ordenadores, con consecuencias revolucionarias. Sustituye a la tradicional fórmula de *datos + algoritmo = programa* por una nueva arquitectura centrada alrededor de una *base de conocimientos* y un *motor de inferencias*, de modo que *conocimientos + inferencia = sistema experto*. Esta fórmula es, obviamente, similar, pero con un enfoque lo suficientemente diferente como para tener profundas implicaciones.



Marcus Wilson-Smith

En busca de consejo
Los sistemas que pueden adquirir conocimientos de expertos y utilizar esos conocimientos para ofrecer consejo o diagnósticos están adquiriendo creciente popularidad en muchas disciplinas, desde la medicina hasta la agricultura y la arquitectura. Los sistemas expertos, diseñados para operar en un campo temático muy delimitado, ofrecen a los profesionales un servicio de consulta en línea para ayudarles en su trabajo

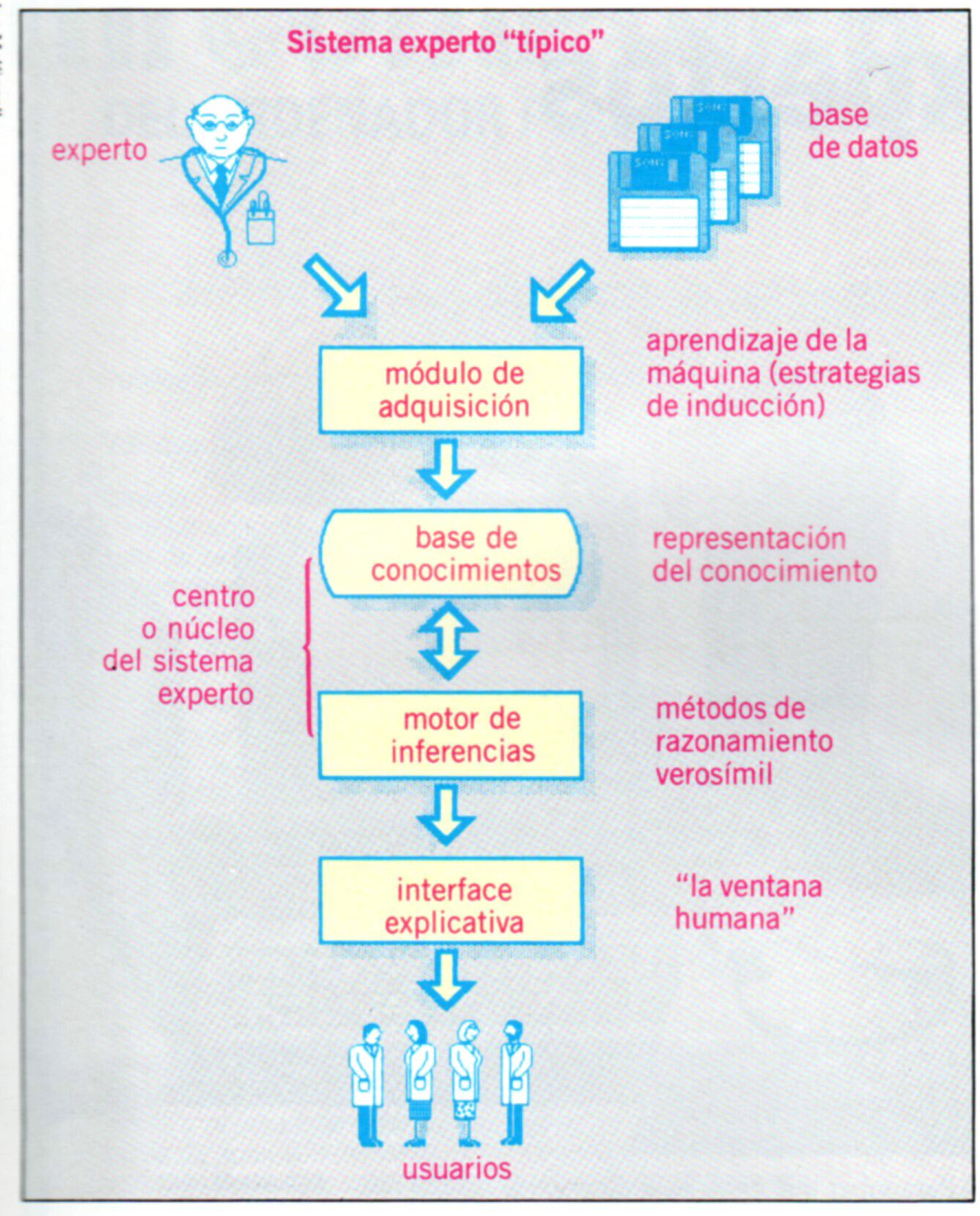
Para saber qué es un sistema experto resulta útil la siguiente lista de verificación de características típicas:

- Un sistema experto se limita a un campo de pericia relativamente delimitado.
- Debe ser capaz de razonar con datos inciertos y reglas no fiables.
- Debe ser capaz de explicar su cadena de razonamiento de una forma exhaustiva.
- Los hechos y los mecanismos de inferencia son "separables": el conocimiento no está codificado como parte de los procedimientos deductivos.
- Está diseñado para crecer por incrementos.
- Típicamente se basa en reglas.
- Su salida es un consejo o sugerencia, no tablas de cifras ni gráficos.

La palabra clave es "conocimiento". Está claro que el objetivo de un sistema inteligente para resolver problemas es omitir la búsqueda ciega o aleatoria. Para hacerlo, un sistema de ordenador ha de explotar la misma ventaja que tiene el experto humano en relación al novato, es decir, la pericia o conocimiento organizado: el conocimiento de hechos, de reglas de inferencia y de estrategias de solución. Un sistema experto totalmente viable tiene cuatro componentes esenciales:

1. La base de conocimientos.
2. El motor de inferencias.
3. El módulo de adquisición de conocimientos.
4. La interface explicativa.

Ian McKinnell



Sinopsis del sistema
Un sistema experto se compone de varios módulos que permiten que el conocimiento pase del experto al usuario final. En primer lugar, el conocimiento se debe adquirir del o de los expertos e incorporar a una base de conocimientos. Para poder hacer predicciones, dar consejo o proporcionar un diagnóstico, el sistema ha de ser capaz de extraer inferencias de la base de conocimientos. Por último, la interface explicativa permite que el usuario se comunique con el sistema con el fin de consultarlo

Los cuatro módulos son críticos. Un sistema basado en el conocimiento puede carecer de alguno de ellos, pero un auténtico sistema experto no puede carecer de ninguno. Hablaremos de estos cuatro módulos de uno en uno y explicaremos cómo trabajan todos juntos.

La base de conocimientos

Los dos componentes fundamentales de un sistema experto son la base de conocimientos y el motor de inferencias. La base de conocimientos almacena información relativa al dominio del tema; sin embargo, la información de una base de conocimientos no es el conjunto pasivo de registros y elementos que usted puede esperar hallar en una base de datos convencional. En cambio, contiene representaciones simbólicas de las reglas de juicio y experiencia de los expertos de una forma que permita al motor de inferencias extraer deducciones lógicas a partir de ellas.

La mayoría de los elementos de una base de conocimientos son no matemáticos. Las dos dificultades fundamentales al desarrollar una base de conocimientos son la representación del conocimiento y la adquisición de conocimientos. El primer problema se refiere a la decisión de cómo codificar el conocimiento de modo tal que el ordenador pueda utilizarlo. En general, se han de representar los elementos siguientes: términos del dominio (la jerga que emplean los expertos de ese campo), relaciones estructurales (las interrelaciones de entidades de los componentes) y relaciones causales (las relaciones causa-efecto entre los componentes).

La tarea del ingeniero de conocimiento consiste en seleccionar los medios adecuados para almacenar simbólicamente tal información. Se han desarrollado cuatro métodos principales:

- Reglas en formato IF...THEN. La condición especifica algún patrón y la conclusión puede ser una acción o aserción.
- Redes semánticas. Éstas representan relaciones entre objetos del dominio (p. ej., la ballena es un mamífero) mediante enlaces entre nudos.
- Marcos. Son estructuras de registros generalizadas que pueden tener valores por defecto y pueden tener codificadas acciones como los valores de ciertos campos o ranuras.
- Cláusulas "trompeta". Ésta es una forma de lógica de predicado en la cual se basa el PROLOG y con la cual el sistema PROLOG lleva a cabo inferencias.

Los primeros sistemas expertos utilizaban casi exclusivamente el formalismo basado en reglas. Una regla de muestra del sistema Mycin para el diagnóstico de infecciones de la sangre es típica de la estructura IF...THEN:

IF:
1. La infección que requiere una terapia es meningitis, y
2. El tipo de infección es fungal, y
3. No se observaron organismos en la solución colorante del cultivo, y
4. El paciente no es un huésped comprometido, y
5. El paciente ha estado en una región donde los coccidiomycosos son endémicos, y
6. La raza del paciente es negra de Asia o India, y
7. El antígeno cryptococal en el csf no fue positivo

THEN
Existen indicios que sugieren que el cryptococo no es uno de los organismos que podrían estar produciendo la infección.

A partir de este ejemplo podemos ver que un sistema experto emplea la jerga técnica del área en el cual fue diseñado para operar; en este caso, la medicina. La construcción IF...THEN utilizada por el Mycin es, esencialmente, una serie de sentencias que se pueden determinar como falsas o verdaderas. Por consiguiente, las sentencias se pueden enlazar mediante operadores booleanos, como AND, para ayudar a su tratamiento por el ordenador. Con el fin de extraer la información requerida para realizar un diagnóstico, el Mycin debe establecer un diálogo con el usuario del sistema. Obviamente, al menos en este sistema, el usuario ha de poseer un cierto nivel de conocimientos en la materia, de modo que pueda comprender las preguntas del sistema experto y responder a ellas.

Los mecanismos de inferencia consisten en métodos de búsqueda o razonamiento que permiten al sistema hallar soluciones y, de ser necesario, proporcionar justificaciones a sus respuestas. Existen dos estrategias de razonamiento globales: el encadenamiento hacia adelante y el encadenamiento hacia atrás.

El encadenamiento hacia adelante implica trabajar hacia adelante a partir de la evidencia (o síntomas) hacia las conclusiones (o diagnósticos). En



un sistema basado en reglas, simplemente implica emparejar las condiciones IF a los hechos, posiblemente por un orden predeterminado. El encadenamiento hacia adelante es fácil de informatizar y adecuado para casos en los que de alguna manera se han de reunir todos los datos. Ejemplos de tales casos son aquellos en los cuales los datos se generan automáticamente mediante un instrumento y en los cuales se ha de rellenar un formulario.

El encadenamiento hacia atrás trabaja desde la hipótesis hasta la evidencia. El sistema elige una hipótesis y busca datos para demostrarla o refutarla. Se puede programar de una forma recursiva y en los sistemas de estilo consultivo conducen típicamente a una clase de diálogo más natural. El problema de qué hipótesis seleccionar en cualquier situación dada aún no está resuelto del todo y, en consecuencia, en la práctica la mayoría de los sistemas emplean una mezcla de encadenamientos hacia adelante y hacia atrás.

Los expertos son notoriamente conocidos por su incapacidad para expresar cómo llegan a sus conclusiones, no necesariamente porque no deseen divulgar secretos sino porque muchos de sus procesos mentales se hallan por debajo del nivel de la consciencia, a nivel intuitivo. De modo que la adquisición de conocimientos ha llegado a considerarse como el principal cuello de botella en el desarrollo de sistemas expertos. Los expertos tienden, no obstante, a ser muy eficientes como críticos. Pueden analizar un caso tomado como ejemplo y decir qué decisión habrían tomado ellos y, si se les solicita, criticar la solución sugerida por un ordenador. Recientemente, por tanto, se ha dedicado mucha atención a desarrollar herramientas de software que permitan a un sistema experto inducir su propio conocimiento a partir de ejemplos previamente clasificados. En efecto, este poder contribuye al proceso de adquisición de conocimientos, superando al mismo tiempo muchas de las dificultades que entraña extraer conocimientos de los expertos humanos y la laboriosa tarea de codificarlos para el ordenador. Aun cuando el sistema no puede hacer todo el trabajo (desde una base de datos de casos a un conjunto de reglas de decisión) por sí mismo, parece útil que pueda refinar su propia base de conocimientos durante un período de "noviciado" o durante el uso, aprendiendo a partir de sus errores. En próximos capítulos analizaremos algunos sistemas de aprendizaje de las máquinas.

La interface explicativa

Una de las cosas buenas del Mycin, el Prospector y muchos sistemas expertos clásicos era que, si se les pedía, podían justificar sus conclusiones. Obviamente, cuando se está poniendo en manos del ordenador la responsabilidad de decisiones de vida o muerte, o incluso de pérdidas o beneficios, es esencial que el sistema explique su razonamiento. He aquí un ejemplo de cómo responde Mycin a una pregunta WHY? (por qué) formulada por su usuario (que es un médico, no un paciente, y no aceptará diagnósticos a ojos cerrados sin más explicación). El diálogo comienza con Mycin solicitando información.

¿Es la meningitis una infección adquirida en el hospital?
WHY?

[2.0] Esto ayudará a determinar los organismos (distintos a aquellos observados en cultivos o frotis) que podrían estar causando la infección. Ya se ha determinado que:
[2.1] la infección que requiere terapia es meningitis, y
[2.2] los organismos no se observaron en la solución colorante de este cultivo csf pendiente, y
[2.3] el tipo de infección es bacteriana, y
[2.4] la neurocirugía no es uno de los elementos relevantes a partir de la historia clínica de este paciente, y
[2.5] la edad de este paciente es de más de 10 días.
Por lo tanto, si:
[2.6] la infección se adquirió mientras el paciente estaba hospitalizado,
entonces existen indicios de que los organismos (distintos a aquellos observados en cultivos y frotis) que podrían estar causando la infección son e. coli (0.75), estafilococo-coag-pos (0.3), pseudomonas-aeruginosa (0.3), klebsiella-pneumoniae (0.5)
—de la [REGLA 545]

Esta facilidad de explicación es en realidad un recorrido parcial a través del proceso de razonamiento del programa, expresado en la jerga médica. Tales explicaciones se pueden producir de forma bastante rápida y económica en sistemas basados en reglas (devolviendo las reglas que se están empleando y sus predecesoras), lo que constituye un punto a favor de la codificación del conocimiento basada en reglas. Observe que las conclusiones llevan unidas estimaciones numéricas. Éstas en realidad no son probabilidades auténticas. Son estimaciones que permiten que el sistema trate con la incertidumbre de una forma coherente y produzca una lista ordenada de diagnósticos probables en el análisis final.

Un buen consejo...



Se entra al Wheat Counsellor a través del sistema Prestel...



La introducción inicial permite que el usuario seleccione entre "consejo de mediada la temporada" o "planificación otoñal"...



El sistema formula entonces al granjero una serie de preguntas sobre cuyas respuestas basará su consejo...



Habiendo interpretado las respuestas del granjero, Wheat Counsellor aconseja en forma de valores de clasificación para los diversos tratamientos químicos disponibles

Cortesía de ICI Plant Protection UK Division

El cerebro del grano
Wheat Counsellor (Consejero del trigo) es un sistema experto desarrollado por ICI y que los agricultores tienen a su disposición, de forma gratuita, a través del Prestel. El sistema está diseñado para aconsejar a los granjeros acerca de los métodos para combatir las enfermedades del trigo, los productos químicos correctos a aplicar y una estimación de las pérdidas probables. Aunque el sistema está diseñado para simular el tipo de conversación que se podría mantener a través del portal de una granja, su conocimiento se ha acumulado a partir de estudios científicos sobre la difusión de las enfermedades de la cosecha. Con el fin de obtener la información a partir de la cual extraerá sus conclusiones, el sistema conduce al granjero a través de una serie de preguntas sencillas. Luego ofrece su consejo en forma de una lista de los tratamientos disponibles, junto con un porcentaje de conveniencia para cada tratamiento

Vista ampliada

Examinemos los archivos sobre los que operan las instrucciones del CP/M

Ya hemos hablado sobre las extensiones de archivos en CP/M, pero ahora vamos a estudiarlas con mayor profundidad. Como hemos podido ver, un archivo CP/M consta de un nombre primario, que puede contener hasta ocho caracteres, seguido por un punto y hasta tres letras, que constituyen una ampliación del nombre del archivo (si bien ésta se puede omitir). Teóricamente, al nombre de archivo primario usted puede añadirle cualquier extensión, pero algunas de éstas están reservadas para fines específicos. Por ejemplo, la extensión COM está reservada para archivos de instrucción (COMmand) CP/M. Esta extensión le informa al CP/M que un archivo con dicha extensión se ha de añadir a la lista de instrucciones transitorias que se pueden ejecutar bajo CP/M.

De modo similar, los programas en BASIC escritos bajo CP/M se deben almacenar con la extensión de archivo .BAS. Muchas versiones de BASIC que se ejecutan bajo CP/M suelen asignarle esta extensión de forma automática a un archivo en BASIC, eliminando la necesidad de que el usuario la digite. Aunque el programa en BASIC se almacena como un archivo fuente (texto), la diferencia entre este archivo y un archivo de texto común es que cuando el programa se vuelva a cargar antes de su ejecución, el CP/M distintiva el programa en BASIC en vez de listarlo como un archivo secuencial ASCII. Sin embargo, muchas máquinas compilan sus programas en BASIC antes de ejecutarlos. Los archivos compilados antes de su ejecución se almacenan con una extensión .INT. Ésta significa que son archivos INTermediarios que se componen de código objeto.

A los programas en código máquina, como a los programas en BASIC, también se les pueden asignar varias extensiones diferentes según cuál sea su estado. Cuando escribimos un programa en assembly, al listado fuente se le debe añadir la extensión .ASM. Si el programa no posee la extensión correcta, el ensamblador residente no intentará ensamblar el programa y simplemente generará un error de archivo.

Cuando intentamos ensamblar el programa, debemos seleccionar una de dos extensiones. Éstas son .HEX o bien .PRN. La extensión HEX significa que lo que se produzca al final del ensamblaje será un archivo objeto HEXadecimal. Por su parte, el ensamblaje con PRN también producirá un listado impreso del ensamblaje con una copia del listado fuente, el código objeto, las direcciones de cada uno de los opcodes y la lista de asignaciones junto con una lista de errores. Por consiguiente, la extensión PRN ofrece una ayuda esencial para la puesta a punto de programas en assembly.

El último grupo de extensiones de archivos que veremos aquí son aquellas relacionadas con los archivos de texto. Muchas versiones de CP/M poseen una pequeña facilidad para edición de textos (a la que se llama ejecutando el archivo de instrucción ED) que se proporciona con el propio sistema operativo, mientras que otras versiones dan por sentado que usted cuenta con un paquete especializado de tratamiento de textos ejecutable bajo CP/M. No obstante, independientemente de la versión que se esté utilizando, las extensiones de los archivos son las mismas.

Aparte de las extensiones de nombres de archivos reservadas que ya hemos mencionado, se pueden utilizar como extensión tres letras casi cualesquiera. En realidad, no es necesario añadir ninguna extensión en absoluto. Si se guarda (SAVE) un archivo de texto sin haberle añadido ninguna extensión, a menudo el CP/M la añadirá por nosotros. Por ejemplo, la utilidad TEXTED añadirá la extensión .TXT al final de un archivo, donde *WordStar* añadiría .$$$.

Archivos de seguridad

Si, tras haber guardado (SAVE) un archivo de texto, usted observa el directorio, verá que se han creado dos archivos: uno con la extensión de archivo que usted ha añadido y otro con el sufijo .BAK. Ésta es una facilidad de seguridad incorporada en el CP/M. Obviamente, es de importancia vital que los documentos importantes no se puedan borrar o corromper accidentalmente, de modo que para evitar hechos tan desastrosos el CP/M creará dos copias de todos los archivos de texto: un archivo normal y un archivo .BAK de seguridad.

De este modo, si accidentalmente usted borra (ERA) un archivo al editarlo, siempre estará disponible la versión de seguridad, gracias a la cual su trabajo anterior no se habrá perdido de forma irrecuperable. Cuando se ha editado y vuelto a guardar un archivo, también se creará una nueva versión de .BAK. Por lo tanto, es una buena idea volver a guardar (SAVE) regularmente los archivos de modo que la copia de seguridad esté siempre actualizada a la última versión.

A pesar de que no es necesario añadir extensiones, una vez implementadas resultan una valiosa herramienta para organizar sus archivos; no sólo en su propio beneficio (para visualizar información sobre la organización de archivos y encabezamientos), sino también para la manipulación de grupos de archivos. Una de las más útiles de estas manipulaciones supone el uso de caracteres de "máscara", que posibilitan el emparejamiento de archivos.

Caracteres de control del CP/M

El CP/M contiene un cierto número de caracteres de control para llevar a cabo muchas de sus funciones. Aunque varios de ellos están ya obsoletos y se implementan muy raramente, se los conserva en el sistema operativo estándar para preservar la compatibilidad entre diferentes ordenadores

CTRL C	Re-carga el disco de sistema CP/M
CTRL C	Inicializa el disco tras su inserción
CTRL M	Termina las instrucciones PIP y devuelve el control al CP/M
CTRL Z	Devuelve control a ED tras oper. de inser.
CTRL P	Envía las entr. a la impr. Volviendo a digitar CTRL P se cancela esta oper.
CTRL U	Borra la línea en curso
CTRL X	Borra la línea en curso y lleva al cursor de vuelva al principio
CTRL M	Ejecuta la línea de instrucción en curso (utilizado en lugar de RETURN)
CTRL H	Retroceso/borrar
CTRL E	Permite entrar 1 línea de instr. larga sin que se ejecute al llegar al final de la 1.ª l.
CTRL R	Repite la línea de instr. en curso

Supongamos, por ejemplo, que un director gerente ha escrito cierto número de memorándums en el mes de julio. Un tiempo después quizá decida que estos archivos ya están obsoletcs y que ya no los necesita en su catálogo. Para suprimir los memorándums de julio puede, por supuesto, recorrer el catálogo y borrar individualmente cada uno de los archivos utilizando la instrucción ERA; pero si el número de archivos es elevado, este proceso será largo y tedioso. Sin embargo, siempre y cuando haya tenido la previsión de darles a todos los memorándums del mes de julio la misma extensión de archivo, pongamos por caso. JUL, este proceso se podrá llevar a cabo con una única instrucción: ERA D:*.JUL (donde D es el nombre de unidad opcional). La colocación del asterisco antes del punto indica al CP/M que ignore el nombre de archivo primario y borre (ERAse) los archivos que lleven la extensión .JUL. Este asterisco también se puede utilizar a la derecha del punto. Empleando asteriscos a ambos lados del punto, la instrucción borrará todos los archivos de un disco, como ERA *.*.

Supresión selectiva

Vamos a suponer que el director gerente conserva todos sus archivos de julio con el sufijo .JUL. Ahora complicaremos el asunto suponiendo, asimismo, que hay una cantidad de documentos importantes que él no desea borrar. En este caso, no cabe lugar al empleo del prefijo*, dado que éste borraría los archivos importantes de julio junto con los memorándums innecesarios. No obstante, nuestro director gerente ha diferenciado sus memorándums de otros archivos otorgándoles los nombres MEM1.JUL,MEM2.JUL, y así sucesivamente. De esta forma todavía es posible borrar todos los memorándums con una sola instrucción sin borrar accidentalmente otros archivos, mediante el uso de la instrucción ERA D:MEM?.JUL. Aquí se está indicando al sistema operativo que borre todos los archivos que comienzan por MEM y terminan con el sufijo .JUL. El ? del cuarto lugar le indica al CP/M que el cuarto carácter no es significativo y que lo puede ignorar.

Este proceso se puede adaptar a cualquiera de las once posibles posiciones en un nombre de archivo. Es particularmente útil en el caso de que nuestro director gerente desee borrar los memorándums tanto de JULio como de JUNio, dado que ambos se reemplazarían por el formato MEM?.JU?. En un nombre de archivo pueden aparecer combinaciones de * y ?, de modo que ERA D:????Q???.* borraría todos los archivos que tuvieran una Q en la quinta posición del prefijo.

El empleo de los caracteres de "máscara" * y ? no está limitado a la instrucción ERA, sino que también se pueden usar con las instrucciones PIP, STAT y REN, entre otras. De modo que, para transferir un número de archivos de un disco a otro, una instrucción típica es PIP B:=A*.*. Esta instrucción buscará todos los archivos de la unidad A que concuerden con el formato (que, en este caso, serían todos ellos) y los copiará en el disco de la unidad B. De esta manera podemos utilizar caracteres de "máscara" para copiar el contenido completo de un disco en otro.

Hasta este momento, a lo largo de la serie hemos hablado de instrucciones que se emplean para controlar el CP/M. Sin embargo, el sistema operativo también hace un uso exhaustivo de caracteres de control para llevar a cabo muchas de sus operaciones. Quizás el carácter de control más útil y más común sea la combinación CTRL-C. La pulsación de estos caracteres producirá una "carga en caliente" del sistema y volverá a cargar el CP/M en el ordenador. Esto es importante, no sólo para volver a cargar el CP/M después de que el sistema quede colgado, sino también al insertar un disco nuevo en una de las unidades.

Recuerde que el CP/M conserva una copia del "enganche" *(log)* de un disco en la memoria del ordenador. Cuando intercambiamos discos en una unidad, debe informarse del cambio al sistema operativo, de lo contrario podría generar errores. Cuando hacemos una carga en caliente del CP/M, éste reenganchará el contenido de cada unidad de disco, lo que nos permitirá continuar. CTRL-C también se puede utilizar para interrumpir la ejecución de muchos programas, dado que el CP/M interrumpirá automáticamente el sistema al reinicializarse a sí mismo.

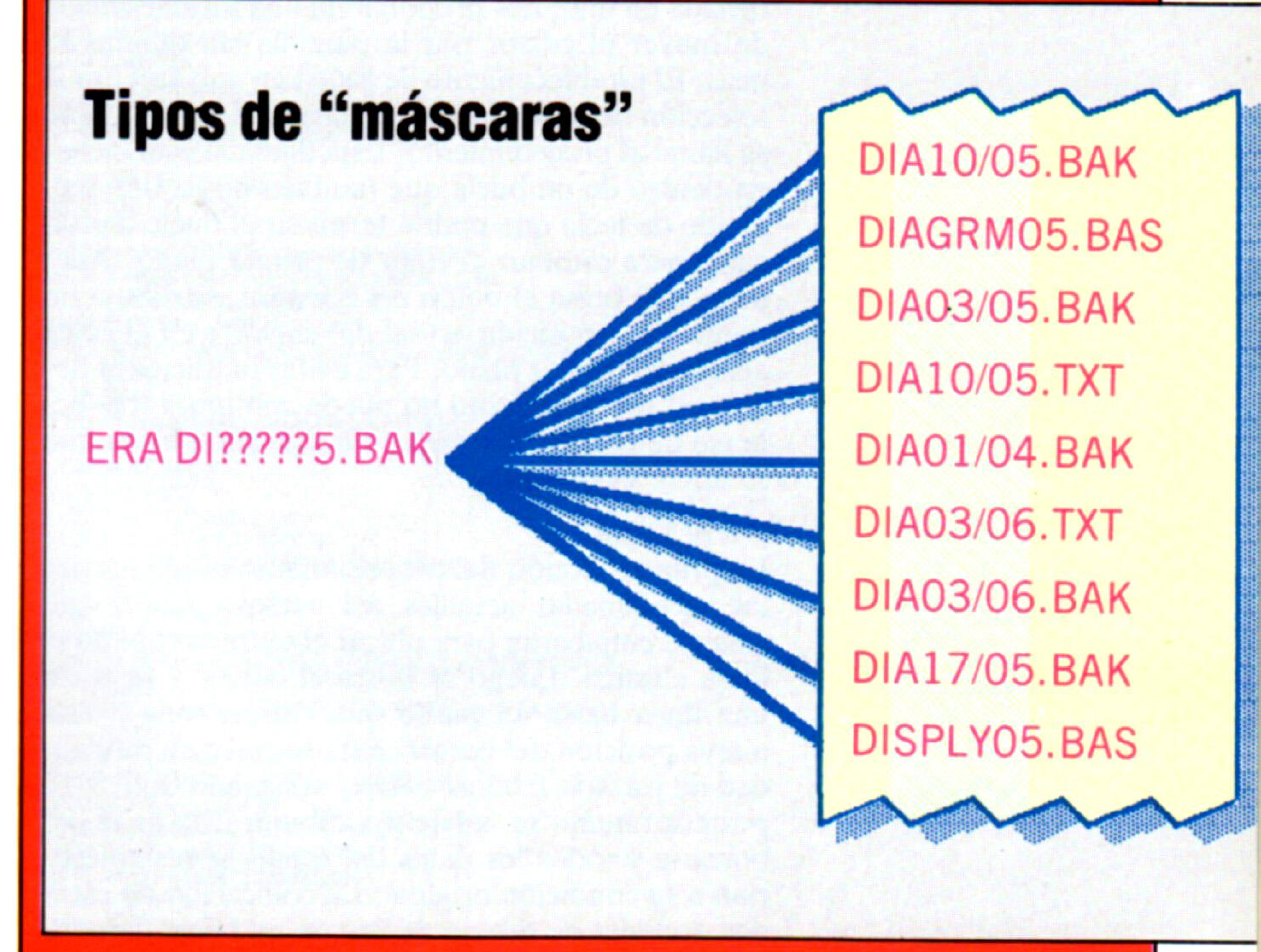
Muchos de los otros caracteres de control están relacionados con la edición de textos y los han adoptado numerosos procesadores de textos que operan bajo CP/M. Por ejemplo, CRTL-H borrará el último carácter, mientras que CTRL-U y CTRL-X borrarán una línea entera. Si se conecta una impresora, se puede enviar texto a la misma digitando CTRL-P. La pulsación por segunda vez de CTRL-P interrumpe esta operación.

El CP/M utiliza otros caracteres de control, muchos de los cuales han quedado obsoletos a raíz de los avances en materia de hardware y software. Existen varias razones por las cuales se han conservado estos caracteres, aun cuando sus funciones parecen haberse duplicado mediante otras teclas; por ejemplo, ahora la mayoría de los ordenadores personales llevan instalada una tecla Delete. Una parte de la explicación es de carácter histórico. Muchos de los primeros microordenadores no tenían tecla Delete o teclas para el cursor, de modo que las facilidades para edición en pantalla debían de estar incorporadas en el sistema operativo.

Otro motivo para conservar el sistema es la compatibilidad. Mientras que, con el correr de los años, los caracteres CTRL y alfabéticos han conservado sus equivalentes ASCII, puede que las teclas más nuevas no los tengan; por ello, para preservar la compatibilidad, se han conservado los caracteres de control. Por último, un usuario que se haya pasado varios años con el CP/M se habrá acostumbrado a usar estas teclas. La alteración del sistema supondría que el usuario debería ahora volver a aprender el control desde el principio.

Habiendo terminado nuestro breve resumen de las instrucciones y archivos tal como están implementados en CP/M, estamos ya en condiciones de llevar a cabo la mayor parte de las funciones cotidianas utilizadas en el sistema operativo.

Usando máscaras
El uso de caracteres de "máscara" es una importante característica del CP/M. Entrando caracteres de "máscara" en posiciones diferentes el ordenador, en efecto, ignorará los caracteres que haya en esa posición cuando busque en el directorio. Ello significa que mediante la cuidadosa asignación de nombres de archivo, el usuario puede manipular listas enteras de archivos clasificados bajo encabezamientos diferentes



Liz Dixon

Garabatos digitales

Finalmente, ofrecemos la segunda parte del programa del trazador para el BBC Micro

La segunda parte del programa del trazador controla el uso de éste en una "modalidad elástica". En lugar de simplemente ofrecer una función de dibujo a mano alzada, cuando se selecciona la modalidad elástica se puede utilizar el trazador para especificar los puntos de origen y final de una línea, el centro y el radio de un círculo, o los tres vértices de un triángulo "sólido". La selección de cada una de estas funciones, junto con una función de trazado de puntos, se realiza pulsando las teclas apropiadas a través del menú visualizado en la parte superior de la pantalla. De este menú se pueden seleccionar, asimismo, las funciones Save, Load y Clear (limpiar la pantalla), similares a aquellas para la modalidad a mano alzada mencionadas en el capítulo anterior. Por consiguiente, esta nueva sección permite crear patrones complejos de una forma muy simple, utilizando el trazador y algunas pulsaciones de teclas. Estos patrones se pueden guardar en disco o cinta para volver a cargarlos de nuevo.

Los 4 procedimientos

- PROCpunto

La llamada a PROCdibujar (procedimiento ofrecido en la primera mitad del programa) con flagart establecida en uno, nos proporciona una forma sencilla de mover el cursor por la pantalla sin dibujar líneas. El establecimiento de flagart en uno asegura la selección de la modalidad lápiz arriba antes de que se llame al procedimiento. Esta llamada está incluida dentro de un bucle que también busca una pulsación de tecla que podría terminar el bucle o utilizarse para cambiar el color de primer plano. Además, si se pulsa el botón del trazador, se dibuja un punto en la posición actual del cursor y en el color actual del primer plano. Para evitar pulsaciones dobles, el procedimiento no puede continuar tras haberse detectado una pulsación del botón hasta que se libere el mismo.

- PROClinea

La primera acción del procedimiento es almacenar las coordenadas actuales del cursor, puesto que éstas se emplearán para ubicar el extremo fijo de la línea elástica. Luego se borra el cursor y se traza una línea hasta un punto que corresponde a una nueva posición del cursor. Ésta se traza en modalidad de trazado Exclusive-OR, utilizando GCOL3. Si posteriormente se volviera a dibujar esta línea, se borraría y todos los datos del fondo se restablecerían a su condición original. La colocación de estas dos acciones de dibujo dentro de un bucle permite al usuario probar varias posiciones de línea, como

Instrucciones del trazador

Menú principal

Seleccionar	Función
1	Modalidad de mano alzada
2	Modalidad elástica

Menú de mano alzada

Seleccionar	Función
S	Guardar pant. en cinta o disco
L	Cargar pant. de cinta o disco
M	Retornar al menú principal
C	Limpiar pantalla
Botón	Lápiz arriba/lápiz abajo

Menú elástico

Seleccionar	Función
S,M,L,C	Igual que en mano alzada
P	Trazar un punto en el color de primer plano actual. M. retorna al menú elástico
D	Dibujar una línea en el color actual de primer plano desde la pos. del cursor al entrar hasta la pos. actual del cursor. Pulsar M o el botón para fijar la línea y retornar al menú elást.
F	Rellenar un triángulo entre 3 puntos especificados mediante puls. del botón
R	Dibujar un círculo en el color de primer plano actual. El centro está dado por la pos. del cursor a la entrada y el radio varía según la pos. actual del cursor. Pulsar M o el botón para dibujar el círculo y retornar al menú elástico

Colores de primer plano

Seleccionar	Función
1	Selecciona el color lógico 1 como color de primer plano (por defecto: rojo)
2	Selecciona el color lógico 2 como color de primer plano (por defecto: amarillo)
3	Selecciona el color lógico 3 como color de primer plano (por defecto: blanco)

Nota: Los colores lógicos se pueden cambiar utilizando VDU 19 para elegir entre los 16 colores disponibles. P. ej., VDU 19,1,6,0,0,0 cambia el color lógico 1 por cyan (color real 6). El color de primer plano se puede volver a seleccionar en cualquier momento



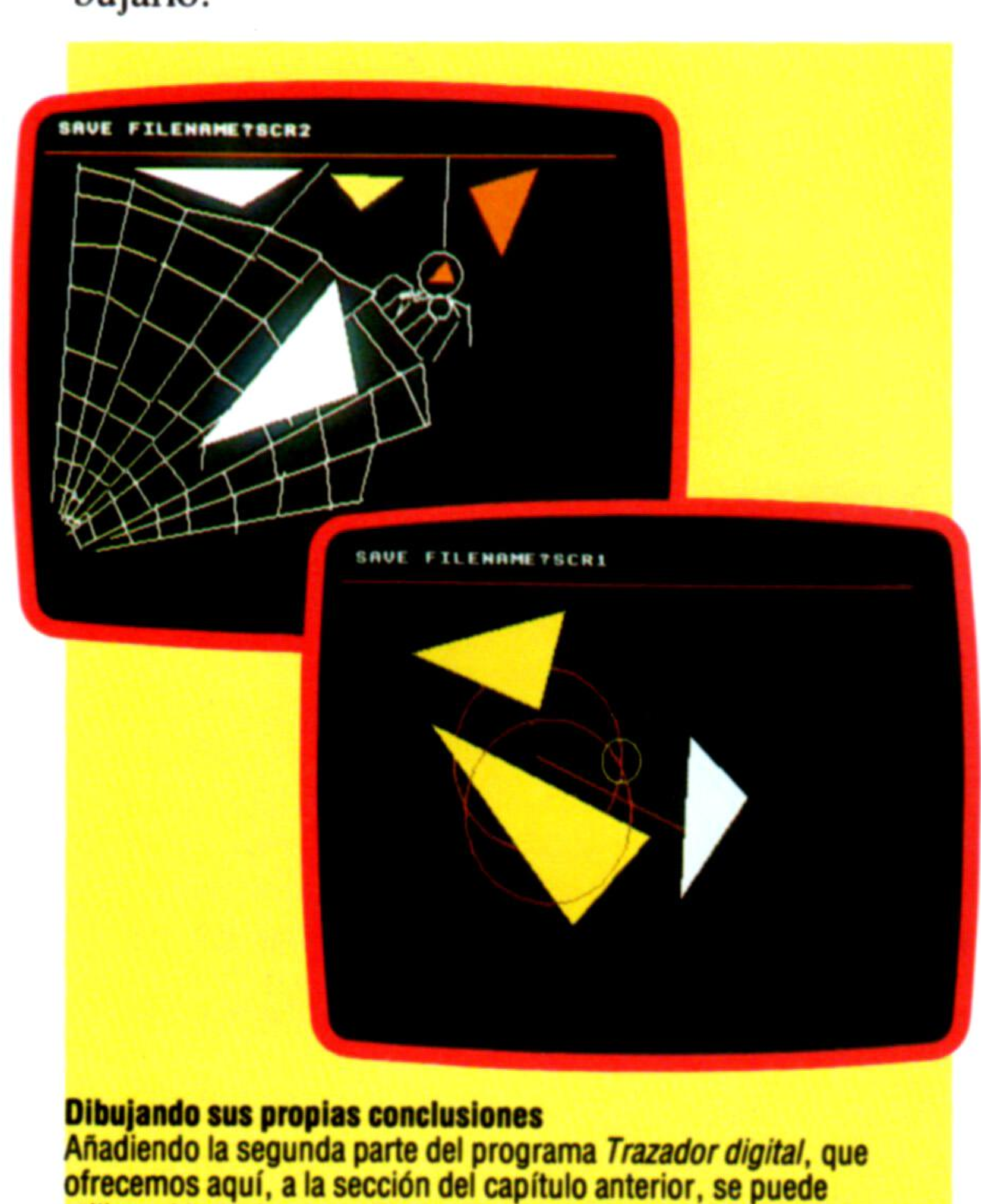
si la línea fuera una banda elástica fijada por un extremo a un punto fijo y por el otro al cursor de pantalla del trazador. Una vez satisfecho con la posición de una línea, el usuario puede "fijar" la línea pulsando el botón del trazador o bien M en el teclado. En este caso el procedimiento omite dibujar la línea una segunda vez, dejándola sin borrar, antes de retornar al menú principal.

- **PROCrellenar**

El procedimiento de relleno saca partido de la facilidad de relleno de triángulos del BBC, a la que se accede mediante el uso de una instrucción PLOT 85,x,y. Esta instrucción toma los dos últimos puntos visitados y el punto especificado en la instrucción como los tres vértices de un triángulo y rellena la forma con un bloque sólido de color. El procedimiento se vale de un método similar al empleado en PROCpunto para permitir que el usuario trace los tres vértices del triángulo a rellenar, utilizando el botón del trazador. Las dos últimas líneas del procedimiento vuelven a visitar los tres puntos para rellenar la forma triangular.

- **PROCcirculo**

La última facilidad que ofrece el programa permite que el usuario especifique el radio de un círculo utilizando el mismo método de banda elástica que el empleado en PROClinea. De hecho, precisamente para este fin se llama a PROClinea desde el procedimiento para dibujar el círculo. Una vez seleccionado el radio, el control retorna a PROCcirculo, que antes que nada debe borrar la línea dejada por PROClinea y borrar el cursor antes de dibujar el círculo. El centro del círculo se retendrá en x1 y x2, las coordenadas del cursor al entrar en el procedimiento, y PROClinea proporciona las coordenadas de un punto de la circunferencia, (x2,y2). Utilizando estos dos puntos se puede calcular el radio del círculo y emplear un algoritmo estándar para dibujarlo.



Dibujando sus propias conclusiones
Añadiendo la segunda parte del programa *Trazador digital*, que ofrecemos aquí, a la sección del capítulo anterior, se puede utilizar el trazador para dibujar líneas rectas, círculos y triángulos "sólidos" para crear visualizaciones en pantalla como éstas

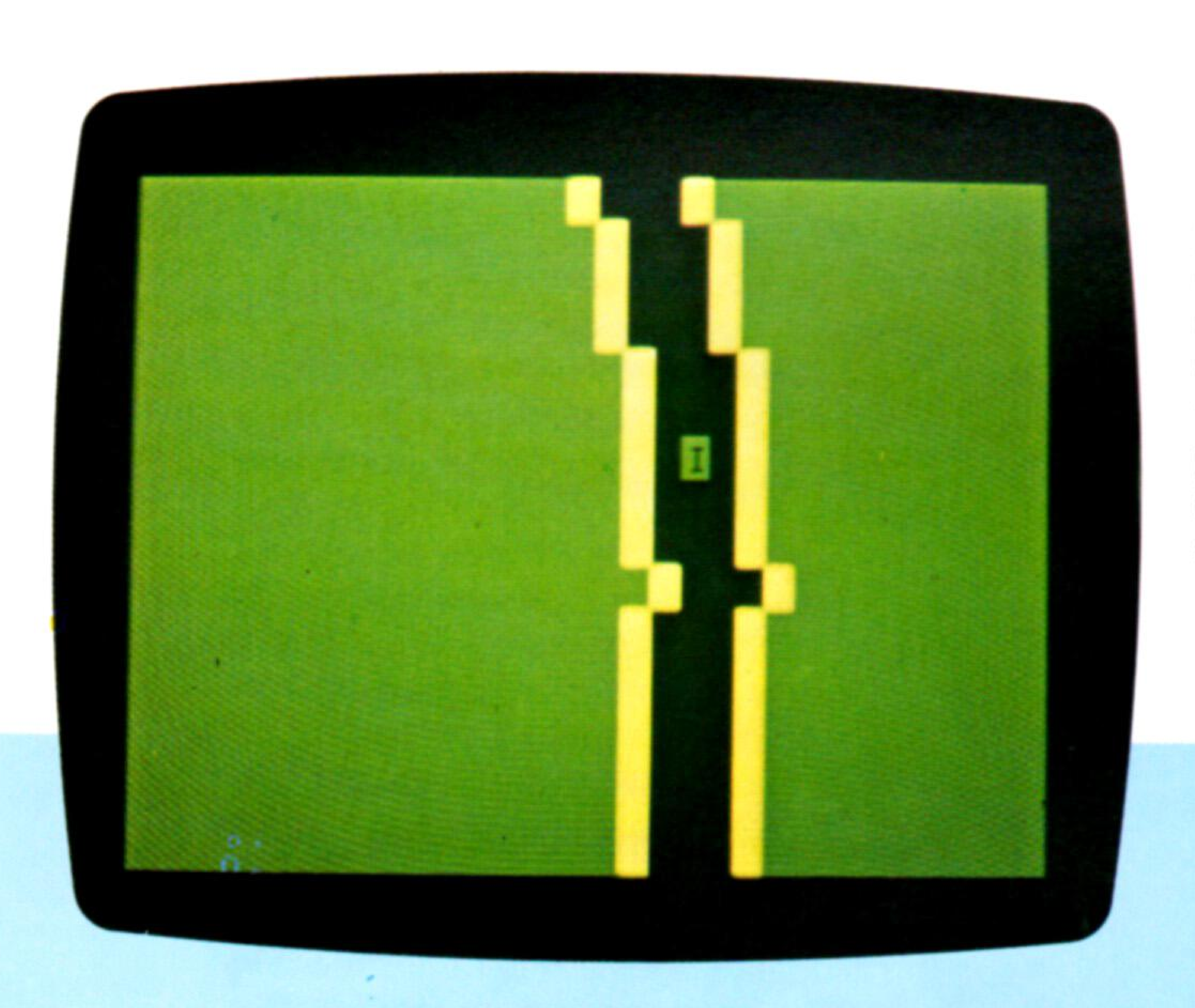
Prog. Trazador digital (II)

El siguiente listado constituye la segunda parte del programa para el trazador digital para el BBC Micro. Se la debe añadir a la primera parte del listado, ofrecida en el capítulo anterior

```
2350 DEF PROCelastica
2360 PROCinforme_elastica
2370 REPEAT
2380 resp$=INKEY$(1):IF resp$<>"" THEN PROCpulsacion_elastica
2390 flagart=1:PROCdibujar
2400 UNTIL flagsalida=1
2410 ENDPROC
2430 DEF PROCinforme_elastica
2440 PROCcalc_xy:antx=x:anty=y:PROCcursor(antx,anty)
2450 PROCreinformar
2460 GCOL 0,1:MOVE 0,920:DRAW 1280,920
2470 ENDPROC
2480 DEF PROCreinformar
2490 PRINT TAB(1,1);SPC(79)
2500 PRINT TAB(1,1);"S=SAVE L=Load M=Menu C=
     Limpiar R=Circulo"
2510 PRINT" D=Dibujar linea   P=Punto   F=Rellenar"
2520 ENDPROC
2540 DEF PROCpulsacion_elastica
2550 IF resp$="C" THEN CLS:PROCinforme_elastica:ENDPROC
2560 IF resp$="M" THEN flagsalida=1
2570 PROCcambiar_color
2580 IF resp$="R" THEN PROCtitulo_circulo:PROCcirculo:PROCinforme_
     elastica:ENDPROC
L.2590,2600
2590 IF resp$="D" THEN PROCtitulo_linea:PROClinea
2600 IF resp$="P" THEN PROCtitulo_punto:PROCpunto:PROCinforme_
     elastica:ENDPROC
2620 IF resp$="S" THEN PROCguardar_pantalla:PROCinforme_
     elastica:ENDPROC
2630 IF resp$="L" THEN PROCcargar_pantalla:PROCinforme_
     elastica:ENDPROC
2640 ENDPROC
2660 DEF PROCtitulo_punto
2670 PRINT TAB(1,1) SPC(79)
2680 PRINT TAB(15,1);"Modalidad punto"
2690 ENDPROC
2710 DEF PROCpunto
2720 REPEAT
2730 flagart=1:PROCdibujar
2740 resp$=INKEY$(1)
2750 PROCcambiar_color
2760 IF (ADVAL(0)AND 3)<>0 THEN PLOT 69,x,y:REPEAT UNTIL
     (ADVAL(0) AND 3)=0
2770 UNTIL resp$="M"
2780 PROCreinformar
2790 ENDPROC
2800 DEF PROCtitulo_linea
2810 PRINT TAB(1,1) SPC(79)
2820 PRINT TAB(15,1);"Modalidad linea"
2830 ENDPROC
2850 DEF PROClinea
2860 X1=x:y1=y:REPEAT
2880 PROCcalc_xy:PROCcursor(antx,anty)
2890 x2=x:y2=y:GCOL 3,colour:MOVE x1,y1:DRAW x2,y2
2900 resp$=INKEY$(1)
2910 IF (ADVAL(0) AND 3)<>0 THEN resp$="M"
2920 PROCcursor(x,y):antx=x:anty=y
2930 IF resp$<>"M" THEN GCOL3,colour:MOVEx1,y1:DRAW x2,y2
2940 PROCcambiar_color
2950 UNTIL resp$="M"
2960 PROCreinformar
2970 ENDPROC
2990 DEF PROCtitulo_rellenar
3000 PRINT TAB(1,1) SPC(79)
3010 PRINT TAB(15,1);"Modalidad rellenar"
3020 ENDPROC
3040 DEF PROCrellenar
3050 FOR I=1 TO 3
3060 REPEAT
3070 flagart=1:PROCdibujar:resp$=INKEY$(1)
3075 IF resp$<> "" THEN PROCcambiar_color
3080 UNTIL (ADVAL(0)AND 3)<>0
3090 PLOT 69.x.y:x(I)=x:y(I)=y
3110 REPEAT UNTIL (ADVAL(0)AND 3)=0:REM ESPERAR SOLTAR
     BOTON
3120 NEXT I
3155 PROCcursor(x,y):REM cursor apagado
3140 FOR I=1 TO 2:PLOT 69,x(I),y(I):NEXT I
3150 PLOT 85,x,y
3160 ENDPROC
3180 DEF PROCtitulo_circulo
3190 PRINT TAB(1,1) SPC(79)
3200 PRINT TAB(15,1);"MODALIDAD CIRCULO"
3210 ENDPROC
3230 DEF PROCcirculo
3240 PROClinea
3250 GCOL 3,colour:MOVE x1,y1:DRAW x2,y2:REM BORRAR
     LINEA
3260 GCOL 0,colour:REM VOLVER A MODALIDAD TRAZADO
     NORMAL
3270 PROCcursor(x2,y2)
3370 radio=SRQ((x2-x1)^2+(y2-y1)^2)
3380 MOVE x1+radio,y1
3390 FOR angulo=0 TO 2*PI STEP 0.1
3400 rx=x1+radio*COS(angulo)
3410 ry=y1+radio*SIN(angulo)
3420 DRAW rx,ry
3430 NEXT angulo
3435 DRAW x1+radio,y1
3440 ENDPROC
```

Autopista

He aquí un ejercicio peligroso, que desaconsejamos a quienes sean poco hábiles. Está escrito en BASIC para el micro Alice, de Matra. No olvide releer las líneas del listado



Usted debe atravesar las cuatro vías de la autopista a una hora en que la circulación es extremadamente densa. Afortunadamente un terraplén central le permite descansar un poco antes de iniciar la segunda parte de la travesía. Cada travesía finalizada con éxito vale un punto. Si usted se estrella, pierde una de sus cinco vidas y el juego se reanuda después de un pequeño intermedio musical. Pulse las teclas W para avanzar y Z para retroceder. Un buen consejo: mire bien a ambos lados antes de entrar en la calzada.

```
5 REM ********************
10 REM *      AUTOPISTA      *
12 REM ********************
13 REM
14 REM INICIALIZACION
15 REM
19 REM RESERVA ESPACIO CADENAS
20 CLEAR 250
30 CLS 0
40 A$=" "
50 B$=" "
60 S=0
70 NP=5
80 P=367
90 P1=P
100 P$=CHR$(191)
110 RESTORE
115 REM INICIALIZACION COCHES
120 FOR I=1 TO 32
130 READ A,B
140 A$=A$+CHR$(A)
150 B$=B$+CHR$(B)
160 NEXT I
170 PRINT @ 64
180 PRINT @ 96
190 PRINT @ 192
200 PRINT @ 288
210 PRINT @ 320
220 PRINT @ P,P$
230 X=RND(31)
240 A$=RIGHT$(A$,X)+LEFT$(A$,32-X)
244 REM
245 REM BUCLE PRINCIPAL
246 REM
250 PRINT @ 352, "VIDA(S) REST.";NP;
255 REM DESPLAZAMIENTO COCHES
260 A$=RIGHT(A$,1)+LEFT$(A$,31)
270 B$=RIGHT(B$,31)+LEFT$(A$,1)
280 PRINT @ 128,B$;
290 PRINT @ 256,A$;
300 PRINT @ 160,A$;
310 PRINT @ 224,B$;
315 REM MOVIMIENTO JUGADOR
320 D$=INKEY$
330 P=P+32*((D$="Z")-(D$="W"))
340 IF P>367 THEN P=367
345 REM RUTA ATRAVESADA?
350 IF P=111 THEN 450
360 C=PEEK(16384+P)
365 REM PERDU?
370 IF C<>96 AND C<>128 AND C<>191
    THEN 600
380 PRINT @ P1," ";
390 PRINT @ P,P$;
400 P1=P
410 D$=INKEY$
420 IF D$<>" " THEN 330
430 GOTO 250
444 REM
445 REM RUTA ATRAVESADA
446 REM
450 PRINT @ P1, " ";
460 PRINT @ P,P$;
470 SOUND 1,1
480 FOR I.0041 TO 200
490 NEXT I
500 PRINT @ P," ";
510 P=367
520 P1=P
530 S=S+1
540 PRINT @ 0, "PUNTOS :";S,
550 PRINT "RECORD :";R
560 GOTO 230
594 REM
595 REM PERDIDO
596 REM
600 NP=NP-1
610 PRINT @ P,CHR$(191);
620 PRINT @ P1," ";
630 GOSUB 1000
640 IF NP=0 THEN 700
650 P=367
660 P1=P
670 GOTO 230
700 IF S>R THEN R=S
710 CLS
720 PRINT    256,"PUNTOS :";S,
730 PRINT "RECORD :";R;
740 PRINT @ 330,"OTRA?";
750 D$=INKEY$
760 IF D$=" " THEN 750
770 IF D$<>"N" THEN 30
780 END
994 REM
995 REM MARCHA FUNEBRE
996 REM
1000 SOUND 15,12
1010 SOUND 15,9
1020 SOUND 15,3
1030 SOUND 15,12
1040 SOUND 55,9
1050 SOUND 45,3
1060 SOUND 45,9
1070 SOUND 15,3
1080 SOUND 15,9
1090 SOUND 1,3
1110 D$=INKEY$
1120 RETURN
2000 DATA 128, 128, 151, 151, 155, 155, 128, 128, 128, 128,
     128, 151, 128, 159
2010 DATA 151, 128, 155, 128, 128, 128, 128, 128, 128, 128,
     151, 128, 155
2020 DATA 128, 128, 128, 128, 147, 151, 155, 155, 128, 128,
     128, 151, 128, 155
2030 DATA 151, 128, 155, 128, 128, 151, 159, 159, 128, 128,
     128, 128, 128
2040 DATA 128, 128, 151, 128, 155, 151, 128, 155
```



El esperado regreso de Atari

La firma Atari hace su reaparición en el mercado con el microordenador 130XE, que se destaca por su elegante diseño y sus 128 Kbytes de memoria



Una nueva imagen
El Atari 130XE es el primer producto que la empresa lanza bajo la dirección del actual presidente, Jack Tramiel. Con 128 K de RAM a bordo y a un precio que sale muy favorecido frente al de muchos ordenadores con sólo la mitad de memoria, Atari confía en que sea ésta la máquina que ayude a la empresa a recuperar su fortuna

A pesar del hecho de que Atari fuera uno de los pioneros en el mercado del ordenador personal, la primera parte de los años ochenta no fue buena para esta firma. En 1984, tras sufrir severas pérdidas, se hizo cargo de Atari el ex director de Commodore, Jack Tramiel, quien se abocó a la tarea de invertir la suerte del achacoso gigante de los ordenadores. Su política de comercializar tecnología de ordenadores al menor precio posible se hizo evidente enseguida en las tiendas, cuando los precios de los micros 600XL y 800XL descendieron sustancialmente de cara al mercado navideño.

Esta jugada no fue suficiente para invertir la tendencia. Atari se enfrentaba a un círculo vicioso de la comercialización de ordenadores: la disminución de las ventas produce falta de apoyo de software, lo que a su vez contribuye a reducir las ventas aún más, lo que significa escasez de fondos para invertir en máquinas nuevas.

No deja de ser irónico el hecho de que fuera Jack Tramiel la persona que más responsabilidad tuvo en colocar a Atari en esa crítica posición. Su agresiva comercialización del Commodore 64 hizo que Atari fuera casi barrida por su rival. A comienzos de 1985 la situación estaba cambiando radicalmente. Commodore se encontraba con una caída de sus ventas y el fracaso del Plus-4 en producir un gran impacto en el mercado, mientras Atari anunciaba un sinnúmero de nuevos productos. La primera de las nuevas máquinas es el Atari 130XE, un ordenador basado en el procesador 6502C.

El 130XE es esencialmente la misma máquina que el ordenador Atari de ocho bits básico, que, en una u otra forma, ha estado en el mercado desde principios de los ochenta. La principal diferencia entre esta máquina y los primeros ordenadores Atari es el elegante nuevo estilo y la cantidad masiva de memoria: el Atari 130XE posee 128 K completos de RAM.

La carcasa del ordenador tiene un aire muy distinto al de sus predecesores. La armazón externa, ligera y de plástico gris, posee el elegante diseño que el público espera de un ordenador moderno, con líneas redondeadas y teclas anchas y esculpidas que facilitan la digitación. Las teclas poseen un recorrido ligeramente mejor que el que ofrecían los modelos anteriores, con la ventaja adicional de no traquetear cuando uno escribe.

Al igual que los otros micros Atari basados en el 6502, el 130XE posee cinco teclas de función preprogramada. Sin embargo, a diferencia de los modelos previos, éstas se han colocado encima del teclado principal en vez de sobre el lado derecho. Ahora están moldeadas en el mismo plástico gris que el resto de la carcasa y están diseñadas en una elegante forma de paralelogramo y no de cuadrados. Éstas representan una gran mejora respecto a las de la serie XL, que eran de metal y su tacto era claramente inestable e inseguro.

Conexiones de interface

Las interfaces instaladas atrás y a la derecha del nuevo Atari también deparan pocas sorpresas. En el costado están las esperadas puertas para palanca de mando tipo D de nueve patillas utilizadas por Atari y adoptadas desde entonces para casi todas las máquinas. En la parte posterior de la máquina está la puerta de control en serie de 13 patillas que emplea Atari para acoplar y enlazar en margarita periféricos tales como aparatos de cassette, unidades de disco e impresoras. A la derecha, insertadas en la carcasa, están las interfaces para cartucho y de ampliación.

La puerta para cartuchos permite a la máquina ejecutar la vasta cantidad de cartuchos de juegos Atarisoft, como *Pacman* y *Galaxians*, que con el correr de los años han sido uno de los puntos fuertes de la empresa.

La puerta para ampliación, sin embargo, se aparta un tanto del estándar anterior. Los primeros ordenadores Atari tenían como bus de ampliación un conector marginal de 50 vías. La máquina nueva tiene instalada como conector de cartuchos un bus mucho más pequeño, de 14 vías. Las interfaces restantes del 130XE son el conector para monitor compuesto, un enchufe hembra RF para aparatos de televisión y el conector para fuente de alimentación Atari estándar.

Una gran memoria

La elegancia del estilo y la compatibilidad están muy bien, por supuesto, pero Atari también las había aplicado a la serie XL sin ningún éxito llamativo. Lo que distingue al 130XL de las primeras máquinas y lo que constituye su evidente punto fuerte de ventas, es la disponibilidad de una gran cantidad de memoria a un precio reducido. Un microprocesador de ocho bits, por supuesto, puede direccionar sólo 64 K de RAM a la vez. Para direccionar el doble de esa cantidad, el ordenador ha de hacer uso del proceso que se conoce como *conmutación de bancos*. Utilizando esta técnica, el ordenador puede "mirar" una ventana de 64 K de los 128 K totales. Si bien la técnica no es perfecta, y aunque las instrucciones extras necesarias para conmutar de un banco de RAM a otro conducen a unas velocidades de recuperación algo más lentas, sí permite que un micro de ocho bits direccione más memoria de la que, en otro caso, sería posible.

En la actualidad la técnica de la conmutación de bancos es más común de lo que cabría pensar. Tanto el Oric Atmos como el Commodore 64 incorporan más de 64 K, y ambos utilizan la conmutación de bancos como un medio para hacer un uso cabal de la memoria disponible.

Puerta para ampliación
La única diferencia del 130XE respecto a las puertas para periféricos de los Atari anteriores es la ranura para ampliación

Conector RF
El control RF está configurado de acuerdo al estándar norteamericano. Por lo tanto, el cable de la antena viene con una caja adicional que contiene la lógica para adecuarse al estándar europeo

Chips de RAM
Los 128 K de RAM están almacenados en estos dos bancos de chips de 8 K

Chip de control de memoria
Este nuevo chip, bautizado como "Freddy", contiene las rutinas de gestión de memoria y conmutación de bancos

Chips de gráficos
Los chips ANTIC y GTIA controlan los gráficos en pantalla del ordenador

CPU
Al igual que todas las máquinas anteriores de Atari, el 130XE está basado en el procesador 6502





Puerta para cartuchos
La puerta para cartuchos permite al ordenador aprovechar la amplia gama de software Atari

Puerta para periféricos
A través de esta interface en serie se pueden conectar entre sí en margarita los periféricos Atari, tales como unidades de disco e impresoras

Puertas para palanca de mando
El ordenador está equipado con un par de puertas para palanca de mando que, naturalmente, responden al estándar Atari

Chip PIA
Un chip 6520 gestiona el control de entrada/salida

Chip de sonido
El chip "POKEY" es responsable de las capacidades de sonido de cuatro octavas del 130XE

Lo que es inusual en el 130XE es su precio. Aunque más caro que el Sinclair Spectrum y el Acorn Electron, el 130XE es considerablemente más económico que los precios de lista del BBC Micro y el Commodore 64. Para poder proporcionar un ordenador de 128 K a ese precio y obtener beneficios, Atari necesita recortar de forma considerable los costos de producción. Hasta cierto punto, la naturaleza de la máquina en sí misma ha mantenido reducido el precio: el 130XE es, esencialmente, una máquina remendada, lo que significa que los costos de desarrollo se han mantenido en un mínimo. Las economías más importantes han tenido lugar en el interior de la máquina.

Recortando costos

El área de memoria comprende 16 chips de RAM de 8 K. El costo de producción de estos chips, que ya no se consideran como un producto de la "tecnología de vanguardia", ha caído dramáticamente en los últimos años y esto se ha reflejado en el precio. Otra forma de reducir los costos es mantener en un mínimo la cantidad de componentes de la placa. Aunque muchos de los chips de la anterior serie XL se han incluido en el 130XE con el fin de preservar la compatibilidad, el trazado de la placa de circuito impreso es excepcionalmente bueno, y ofrece un aspecto mucho menos abarrotado que el de muchas máquinas cuya capacidad de memoria es de apenas la mitad.

Por último, la firma Atari ha hecho fuertes inversiones en plantas de ensamblaje automatizadas, de las cuales el 130XE es el primer producto. Todos los componentes de la placa se sueldan mediante máquinas.

Dado que, en esencia, no se ha alterado ninguno de los chips de ROM, de gráficos y de sonido, para el usuario el ordenador es exactamente el mismo que los modelos anteriores, con los excelentes gráficos y sonido que son propios de Atari. Uno de los cambios fundamentales que redundan en beneficio del usuario es el manual. Los manuales de consulta que acompañaban a los modelos anteriores estaban simplificados hasta el punto de que parecían estar dirigidos a niños. El tutor de BASIC está muy mejorado y la empresa ha ofrecido en el apéndice algunas especificaciones técnicas. No obstante, para una explicación completa sobre el dialecto, continúa siendo necesario adquirir el Manual de Referencia del BASIC Atari.

Aunque la gama de micros Atari reclama una mejora urgente, la llegada del 130XE es en cierto modo un enigma. Los 64 K extras de RAM le proporcionan al programador muchísima más memoria, pero sin embargo aún no hay ningún programa disponible para sacar partido de ellos, aun teniendo en cuenta la compatibilidad con el software Atari ya existente en el mercado. Normalmente, uno esperaría que el lanzamiento de una máquina como ésta fuera una maniobra preliminar a una seria arremetida destinada al mercado de pequeña gestión. Sin embargo, la nueva dirección de Atari ha negado taxativamente que tuvieran esas intenciones. Quizá la verdadera razón del lanzamiento del 130XE es que Atari pretendía anticiparse al lanzamiento del Commodore 128, una máquina compatible con el Commodore 64 que también posee memoria extra pero cuyo precio es más elevado.

ATARI 130XE

DIMENSIONES
350×233×63 mm

CPU
6502C operando a 1.79 MHz

MEMORIA
128 K de RAM, 24 K de ROM

PANTALLA
Visualización de textos de 40×24, 320×192 pixels (alta resolución) con 256 colores disponibles

INTERFACES
Puerta para cartuchos, enchufe TV, conector para monitor compuesto, dos puertas para palanca de mando, puerta de entrada/salida en serie, interface para ampliación

LENGUAJES DISPONIBLES
BASIC Atari, LOGO, FORTH, PILOT

TECLADO
62 teclas, incluyendo cinco teclas de función preprogramadas

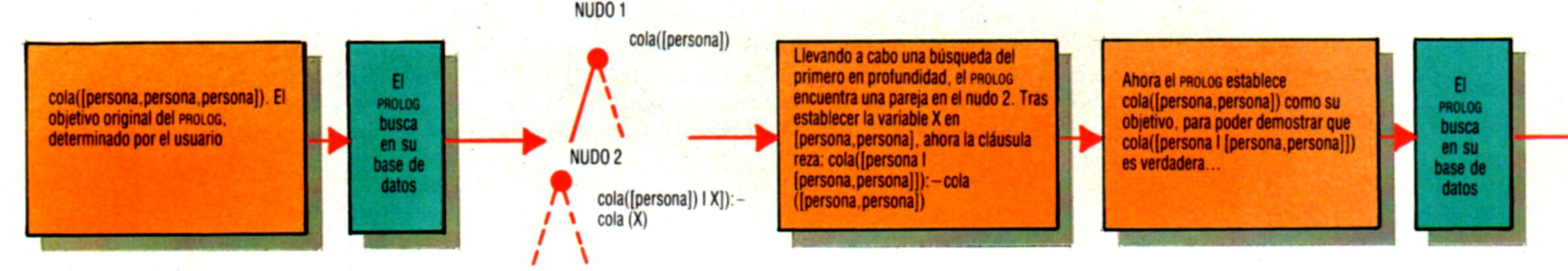
DOCUMENTACION
El manual ofrece una explicación completa sobre el BASIC Atari, si bien el tono general sigue siendo de una simplificación excesiva. En el apéndice se ofrecen explicaciones sobre las características de las interfaces y cómo conmutan los otros 64 K de RAM

VENTAJAS
El 130XE posee la ventaja de ofrecer los puntos fuertes ya tradicionales de Atari, y posee 64 K adicionales de RAM que se pueden utilizar como un "disco de silicio" para almacenamiento y recuperación rápidos

DESVENTAJAS
El ordenador no resuelve el problema básico de Atari, el de carecer de una amplia base de software de firmas independientes. El hecho de que el 130XE sea esencialmente una versión remozada de una máquina que lleva ya varios años en el mercado, podría indicar que no generará el interés que tanto necesita la empresa para reafirmar su futuro financiero



Haciendo cola



Búsqueda profunda

A la hora de preguntar
Este diagrama de flujo muestra al PROLOG en acción, respondiendo a una sencilla pregunta formulada por el usuario. Observe que durante la ejecución del programa, a la variable X se le dan dos valores diferentes, manteniendo, no obstante, ambos valores. Esto es posible porque el PROLOG trata a las variables como locales de cada invocación separada de una cláusula

Mientras que lenguajes como el BASIC y el PASCAL poseen flujos de control secuenciales, pasando el control de sentencia en sentencia por estricto orden de arriba abajo (a menos que un bucle o un GOTO lo interrumpa), en PROLOG el flujo de control asume la forma de una búsqueda del primero en profundidad a través de las cláusulas del programa.

Pensemos en el programa como un árbol, con el objetivo (proposición) a demostrar en la raíz y todos los subobjetivos como puntos de elección donde se dividen las ramas inferiores. Existen muchas formas de buscar en un árbol como éste, pero el método que emplea el PROLOG consiste en tomar la rama situada más a la izquierda y seguirla hasta la mayor profundidad posible. Mientras va probando cada rama, marca su recorrido, y cuando llega abajo y no puede seguir adelante, retrocede hasta el punto de elección más próximo, tomando la rama situada más a la izquierda *que aún no haya tomado* y continúa su avance a partir de allí. De esta forma, el sistema explorará todos los caminos a través del árbol y habrá intentado todas las formas posibles de demostrar el objetivo situado arriba de todo.

Una búsqueda del primero en profundidad que sea exhaustiva, ofrece plenas garantías de cubrir todos los caminos, pero puede llegar a ser un asunto muy largo. De hecho, el PROLOG consigue ahorrarse a sí mismo algo de trabajo. Una cláusula del PROLOG es, como ya hemos visto en el capítulo anterior, una regla que dice que el objetivo del encabezamiento es verdadero y los subobjetivos también:

```
objetivo:–subobjetivo1,subobjetivo2,
subobjetivo3... etc.
```

y así sucesivamente. Probablemente esto se comprenda mejor así:

```
IF subobjetivo1 es verdadero
   AND subobjetivo2 es verdadero
   AND subobjetivo3 es verdadero
   AND etc.
THEN objetivo es verdadero.
```

Dado que los objetivos están relacionados entre sí mediante AND, toda la cláusula fracasará si no se puede demostrar alguno de ellos. De modo que el PROLOG va trabajando a través de los subobjetivos de izquierda a derecha y, si no consigue demostrar alguno, abandona en ese punto y ya no se preocupa por el resto.

El retroceso puede hacer que los programas se comporten de una forma que haga que el orden por el cual están escritas las instrucciones sea casi irrelevante. Sin embargo, la ventaja es que el flujo de control, en vez de ser una cuestión fundamental como lo es en BASIC, tiene una importancia sólo menor, dejando que usted se concentre en la estructura lógica de su problema.

El hincapié que hace el PROLOG en una sentencia "declarativa" del problema no significa que usted no pueda ver comportarse a sus programas de forma procesal. La cláusula en PROLOG:

```
marciano(X):–num. de extremidades
  (X.7),num. de cabezas (X.2),
  sabe programar en (X,cobol).
```

se puede leer declarativamente como: "X es un marciano si X posee 7 extremidades, dos cabezas y sabe programar en COBOL". Su lectura procesal sería: "Para demostrar que X es un marciano, demostrar primero que posee siete extremidades, luego, que posee dos cabezas, luego, que sabe programar en COBOL".

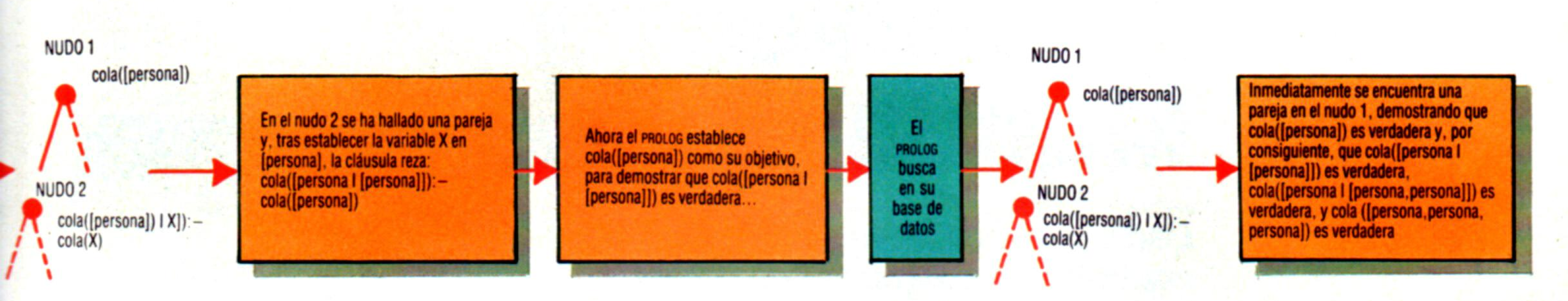
El PROLOG no tiene una "tipología" tan severa como la mayoría de los otros lenguajes. Tampoco es exigente en cuanto a los tipos de datos de los argumentos de sus términos. De modo que el término pred(Argumento) se podría utilizar en ocasiones diferentes con la variable Argumento establecida como entero, como un "átomo" (como marty, venusiano, d24, etc.), o como una lista. No obstante, el PROLOG trata a sus tipos de datos de formas diferentes, permitiendo, por ejemplo, manipular números aritméticamente.

Quien esté familiarizado con el LOGO o el LISP ya se habrá encontrado antes con el tipo de datos lista y las formas especiales que se emplean para manipular listas. En PROLOG una lista se escribe encerrada entre corchetes con los elementos de la lista separados mediante comas. De modo que [manzana,pera,banana] es una lista de frutas, [a,f,e,g,r,x] es una lista de letras y así sucesivamente.

Para penetrar en las listas, el PROLOG nos permite separar un elemento por vez mediante la supresión del primer elemento. La notación [Encabezamiento/Cola] describe una lista con el elemento Encabezamiento y el resto de la lista en la lista Cola. La aplicación de "I" a nuestra lista de frutas nos dará [manzana I [pera,banana]]. Como puede ver, las listas pueden tener como miembros a otras listas. Como caso especial, si tomamos la lista [z80] con un elemento y la descomponemos con I, obtenemos [z80 I []]. La lista de la cola es [], que representa la lista vacía.

El PROLOG permite la recursión que, en realidad, es el estilo normal de un programa en PROLOG. Una definición recursiva es aquella que define algo en términos de sí mismo. En PROLOG podríamos escribir:

```
cola([persona]).
cola([persona I X]):–cola(X).
```



Kevin Jones

Cuando hay dos o más cláusulas, como tenemos aquí, que poseen el mismo encabezamiento, se dice que se trata de un *procedimiento*. El procedimiento cola tiene una cláusula que define a una cola como una lista que posee un elemento, persona. Luego posee una segunda cláusula que nos informa que una cola también podría ser una lista con el elemento persona en el encabezamiento y con una lista llamada X como su cola. Luego, a partir del lado derecho de esta cláusula, vemos que X debe ser una cola. Si le damos al PROLOG un objetivo como:

```
cola([persona,persona,persona]).
```

solicitándole que nos diga si la lista [persona,persona,persona] es una cola, primero busca en su base de datos una cláusula para emparejar con nuestro objetivo. La primera que hallará será cola([persona]). Ésta no concuerda porque las listas no son idénticas, de modo que seguirá explorando hacia abajo hasta la cláusula siguiente: cola([persona I X]):–cola(X). Ésta tampoco concuerda y rellena los valores de las variables de este modo:

```
cola([persona I [persona,persona]]):–
   cola([persona,persona]).
```

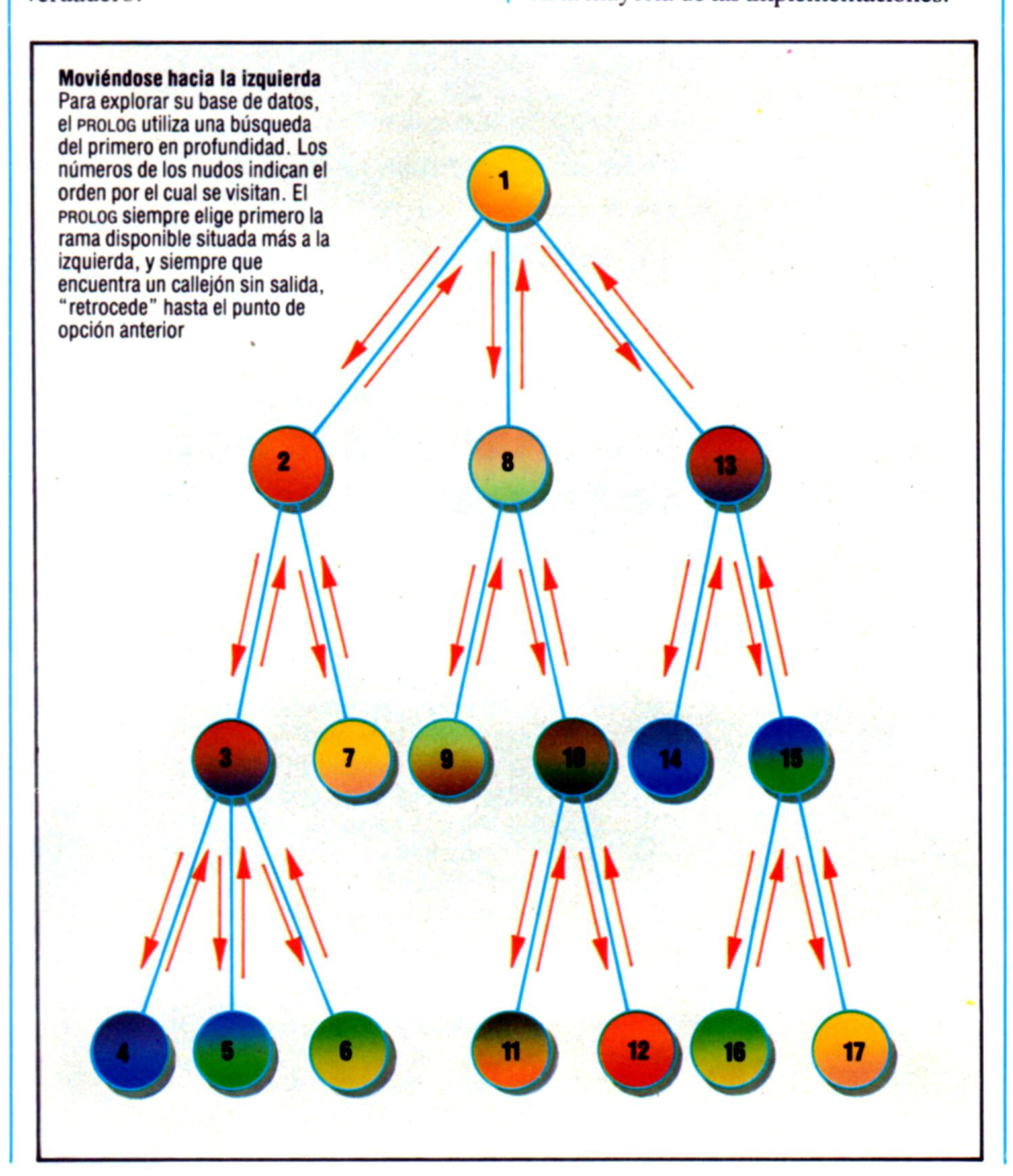
Para demostrar que el objetivo del encabezamiento es verdadero, debe demostrar que el subobjetivo lo es. De modo que el PROLOG toma cola([persona,persona]) como su objetivo y comienza a explorar hacia abajo las cláusulas *desde arriba* para hallar una pareja. Nuevamente, cola([persona]) no concuerda, pero la segunda cláusula sí:

```
cola([persona I [persona]]):–
   cola([persona]).
```

Un punto importante que se debe notar aquí es que la variable X, que en la primera ejecución se estableció en [persona,persona], ahora se ha establecido en [persona]. Pero se ha conservado el primer valor de X. Esto es posible porque las variables son locales a cada invocación separada de una cláusula; por tanto, se puede pensar en cada llamada a cola([persona I X]) como si se empleara una variable única y separada. En PROLOG no existe nada que se parezca a una variable global.

Ya emparejado el encabezamiento de este objetivo, para demostrarlo necesitamos demostrar el subobjetivo a la derecha del símbolo :–, que es cola([persona]). Se inicia otra exploración de la cláusula base y esta vez se encuentra una pareja con cola ([persona]), que encaja directamente y significa que ahora cola([persona, persona]) (nuestro objetivo previo) resulta ser verdadera. Esto a su vez significa que cola([persona,persona, persona]), el objetivo original, es verdadero.

El procedimiento cola nos muestra varias cosas importantes sobre el PROLOG. Por ejemplo, que la disposición de las cláusulas puede ser crucial. (Pruebe colocar las dos cláusulas en el orden contrario y vea lo que sucede al intentar demostrar un objetivo.) Asimismo, ilustra cómo dar cláusulas adicionales es lo mismo que dar formas alternativas de proporcionar un objetivo, tal como si hubiéramos utilizado un OR lógico entre las cláusulas. Ello significa que el PROLOG no necesita de un operador OR, si bien éste se suministra en la mayoría de las implementaciones.



Moviéndose hacia la izquierda
Para explorar su base de datos, el PROLOG utiliza una búsqueda del primero en profundidad. Los números de los nudos indican el orden por el cual se visitan. El PROLOG siempre elige primero la rama disponible situada más a la izquierda, y siempre que encuentra un callejón sin salida, "retrocede" hasta el punto de opción anterior

Kevin Jones

El Nuevo Mundo (I)

Dedicamos los últimos capítulos a considerar los "complementos al BASIC" necesarios

El programa se escribió para el Commodore 64, pero utilizando un BASIC mínimo allí donde ello es posible. Los problemas de conversión para ejecutar el programa en el Spectrum quedan comprendidos en dos áreas principales: primero, el Spectrum sólo permite utilizar nombres de variable de letra única en matrices o contadores de bucles FOR...NEXT. Ofrecemos aquí una tabla de conversión. En segundo lugar, la manipulación de series en el Spectrum es inusual en tanto y en cuanto no se dispone de LEFT$, MID$ ni RIGHT$, si bien todas poseen un equivalente en el Spectrum. Los usuarios del Spectrum habrán de referirse, para estas conversiones, a los complementos que hemos ido ofreciendo con cada módulo. Además, los PRINT CHR$(147) se han de reemplazar por CLS, y las líneas que aguardan pulsaciones de tecla con la forma:

```
<n.º línea>GET I$:IF I$=""THEN<n.º línea>
```

se han de reemplazar por:

```
<n.º línea>LET I$=INKEY$:IF I$="" THEN
GO TO <n.º línea>
```

En el próximo capítulo ofreceremos la segunda parte del listado completo de este programa.

Conversión de variables para el Spectrum

Equivalente Microsoft	Finalidad	Spectrum
TS(,)	Categoría/fortaleza trip.	T(,)
WG()	Tasas salariales	W()
CC()	Contadores categ. de trip.	Y()
PA()	Provisiones adquiridas	A()
PC()	Costo de las provisiones	C()
PN()	Necesidades aprovision.	N()
OC()	Costos merc. a comerciar	O()
OA()	Cant. merc. a comerciar	G()
HR()	Indic. de medias raciones	H()
RR()	Indicadores de eventos	(R)
AO()	Cant. merc. comerciadas	E()
EQ(,)	Valores de intercambio	Q(,)
V1()	Valores al zarpar	B()
V2()	Valores al regresar	D()
S1	Cont. reducción fortalezas	S
S3	Contador impresión lenta	S
S4	Contador breve demora	J
S5	Contador larga demora	S

```
1 REM ****************
2 REM **    Juego     **
3 REM **  Mercantil   **
4 REM ** Nuevo Mundo  **
5 REM ****************
6 :
```

La primera sección del programa inicializa las variables y matrices que se requerirán después

```
9 K$="    PULSE CUALQUIER TECLA PARA CONTINUAR"
10 DIM TS(16,2):REM TIPO/FORTALEZA DE LA TRIPULACION
11 CN=0:REM NUMERO DE TRIPULANTES
12 MO=2000: REM DINERO INICIAL
13 DIM WG(5):WG(1)=10:WG(2)=25:WG(3)=15:WG(4)=20
   :WG(5)=15:REM SALARIOS
14 WT=0:REM FACTURA SALARIAL SEMANAL
15 CM=16:REM TRIPULACION MAX
16 DIM C$(5):C$(1)="MARINERO":C$(2)="MEDICO":
   C$(3)="MECANICO"
17 C$(4)="OFICIAL":C$(5)="COCINERO"
18 DIM CC(5):REM CONTADOR DE CADA TIPO DE TRIPULACION
19 REM **** MATRICES DE APROVISIONAMIENTO ****
20 DIM U$(4):U$(1)="KILO":U$(2)="KILO":U$(3)="KILO":
   U$(4)$"BARRIL"
21 DIM P$(4):P$(1)="VEG":P$(2)="FRUTA":P$(3)="CARNE":
   P$(4)="AGUA"
22 DIM PA(4)
23 DIM PC(4):PC(1)=.5:PC(2)=1:PC(3)=2:PC(4)=.5
24 DIM PN(4): PN(1)=2:PN(2)=1:PN(3)=1:PN(4)=.5
30 DIM OA(6)
31 DIM D$(6)
32 D$(1)="FRASCO DE MEDICINA":D$(2)="ARMA":D$(3)="BOLSA DE
   SAL"
33 D$(4)="BALA DE TELA":D$(5)="CUCHILLO":D$(6)="JOYA"
34 DIM OC(6)
35 OC(1)=1:OC(2)=5:OC(3)=.2
36 OC(4)=2:OC(5)=.5:OC(6)=1
40 JL=8:REM DURACION DEL VIAJE
41 EW=0: REM SEMANAS EXTRAS
42 DIM RR(16)
43 REM INDICADORES PARA MOSTRAR SI YA HA OCURRIDO EL EVENTO
   AZAR(N)
44 RC=0
45 REM CONTADOR EVENTO AL AZAR HASTA AHORA
46 RM=13
47 G$="N":REM INDICADOR DE BUEN TIEMPO PARA USAR CON FACTOR
   MOTIN
48 A$="N":B$="N"
49 DIM M(6):REM SEÑALA SI YA SE HAN PRODUCIDO LOS EVENTOS
   MAYORES
60 DIM T$(3):T$(1)="PERLAS":T$(2)="FIGURILLAS":T$(3)
   ="ESPECIAS"
61 DIM V1(3):V1(1)=2:V1(2)=2:V1(3)=1
62 DIM V2(3):V2(1)=2+(INT(RND(1)*1)/2):V2(2)=2+(INT(RND(1)*
   3)-1)
63 V2(3)=2+(INT(RND(1)*1)/2)
64 DIM EQ(4,3)
65 EQ(1,1)=0.5:EQ(1,2)=0.5:EQ(1,3)=1
66 EQ(2,1)=5:EQ(2,2)=5:EQ(2,3)=10
67 EQ(3,1)=3:EQ(3,2)=3:EQ(3,3)=6
68 EQ(4,1)=2:EQ(4,2)=2:EQ(4,3)=4
69 DIM AO(3)
80 PRINT CHR$(147):S$="       JUEGO MERCANTIL NUEVO
   MUNDO*":GOSUB 9100:PRINT
81 GOSUB 9200
82 S$="ERES EL CAPITAN DE UN BARCO*":GOSUB 9100:PRINT
83 S$="QUE SE DIRIGE AL NUEVO MUNDO.SE*":GOSUB 9100:PRINT
84 S$="CALCULA QUE EL VIAJE DURARA OCHO*":GOSUB 9100:PRINT
85 S$="SEMANAS, PERO PODRIA DURAR MAS. DEBES*":GOSUB
   9100:PRINT
86 S$="CONTRATAR UNA TRIPULACION, PAGARLES,
   COMPRAR*":GOSUB 9100:PRINT
87 S$="PROVISIONES, EQUIPO Y MERCANCIAS*":GOSUB 9100:PRINT
88 S$="PARA COMERCIAR. DISPONES DE 2000 PIEZAS*":GOSUB
   9100:PRINT
89 S$="DE ORO PARA GASTOS.*":GOSUB 9100:PRINT:GOSUB 9200
90 PRINT:S$="       BUENA SUERTE!*":GOSUB 9100:GOSUB 9200:PRINT
91 S$=K$:GOSUB 9100
94 GET P$:IF P$=""THEN 94
95 GOSUB 9200
```

Esta sección llama a las subrutinas que permiten que el jugador contrate una tripulación, provea al barco de provisiones para el viaje y adquiera las mercancías a comerciar en el Nuevo Mundo

```
500 GOSUB 1000
550 GOSUB 2000
600 GOSUB 3000
605 REM *** LISTO PARA EMPEZAR ****
610 PRINT CHR$(147)
615 S$="AHORA ESTAS LISTO PARA EMPEZAR*":GOSUB 9100
625 S$="EL VIAJE.*":GOSUB 9100
630 GOSUB 9200
635 PRINT:S$="CUENTAS CON LA SIGUIENTE TRIPULACION:*":GOSUB
    9100
640 GOSUB 9200
645 FOR T=1 TO 5
650 IF CC(T)=0 THEN 670
655 PRINT CC(T);
```



```
660 PRINT C$(T);
662 IF CC(T)=1 THEN PRINT"    ":GOTO 668
664 PRINT "S"
668 GOSUB 9200
670 NEXT
674 GOSUB 9200
675 PRINT:S$="Y LAS SIGUIENTES PROVISIONES:*":GOSUB 9100
680 GOSUB 9200
685 FOR T=1 TO 4
690 IF PA(T)=0 THEN 710
695 PRINT PA(T);U$(T);"S DE";
700 PRINT P$(T)
708 GOSUB 9200
710 NEXT
715 GOSUB 9200
720 PRINT:S$="  TAMBIEN POSEES:*":GOSUB 9100
725 GOSUB 9200
730 IF OA(1)=0 THEN 740
733 IF OA(1)=1 THEN S$="FRASCO DE MEDICINA"*:GOSUB 735
734 S$="FRASCOS DE MEDICINA*"
735 PRINT OA(1);:GOSUB 9100
736 GOSUB 9200
740 IF OA(2)=0 THEN 750
743 IF OA(2)=1 THEN S$="ARMA*":GOTO 745
744 S$="ARMAS*"
745 PRINT OA(2);:GOSUB 9100
746 GOSUB 9200
750 IF OA(3)=0 THEN 760
753 IF OA(3)=1 THEN S$="BOLSA DE SAL*":GOTO 755
754 S$="BOLSAS DE SAL*"
755 PRINT OA(3);:GOSUB 9100
756 GOSUB 9200
760 IF OA(4)=0 THEN 770
763 IF OA(4)=1 THEN S$="BALA DE TELA*":GOTO 765
764 S$="BALAS DE TELA*"
765 PRINT OA(4);:GOSUB 9100
766 GOSUB 9200
770 IF OA(5)=0 THEN 780
773 IF OA(5)=1 THEN S$="CUCHILLO*":GOTO 775
774 S$="CUCHILLOS*"
775 PRINT OA(5);: GOSUB 9100
776 GOSUB 9200
780 IF OA(6)=0 THEN 790
783 IF OA(6)=1 THEN S$="JOYA*":GOTO 785
784 S$="JOYAS*"
785 PRINT OA(6);:GOSUB 9100
786 GOSUB 9200
790 GOSUB 9200
792 PRINT:PRINT"TE QUEDAN  ";MO;"  PIEZAS DE ORO"
796 GOSUB 9200
797 S$="PULSA CUALQUIER TECLA PARA COMENZAR EL VIAJE*"
798 GOSUB 9100
799 GET P$:IF P$=""THEN 799
800 WT=0:REM PONER A CERO TOTAL SALARIOS
801 H$="N"  :REM INDICADOR DE MEDIA RACION
802 DIM HR(4):HR(1)=1:HR(2)=1:HR(3)=1:HR(4)=1
```

Aquí comienza el bucle principal, utilizando WK para descontar las semanas transcurridas

```
820 FOR WK=1 TO JL : REM BUCLE PRINCIPAL DEL VIAJE
825 GOSUB 4000:REM INFORME ESTADO TRIPULACION
830 GOSUB 4200:REM INFORME PROVISIONES
835 GOSUB 4300:REM INFORME OTRAS MERCANCIAS
840 GOSUB 9200:PRINT CHR$(147)
842 PRINT:PRINT:PRINT
843 S$="SE CALCULA QUE EL VIAJE*":GOSUB 9100
844 PRINT"DURARA AUN OTRAS";INT(JL-WK+1);SEMANAS   "
845 GOSUB 9200
846 PRINT:S$=K$:GOSUB 9100
847 GET I$:IF I$="" THEN 847
850 GOSUB 5000:REM COMPROBAR FACTURA SALARIAL
855 GOSUB 5100:REM REPARTIR RACIONES
860 GOSUB 5500
861 REM GO TO GENERAR EVENTO AL AZAR
870 GOSUB 6500:REM IR A EVENTO MAYOR
875 IF HR(3)=.5 AND RND(1)<.5 THEN PRINT CHR$(147):GOSUB 6050
878 REM ALBATROS SI ESCASEA LA CARNE
879 GOSUB 7200
880 GOSUB 5300:REM INFORME DE FINAL DE SEMANA
889 NEXT WK
890 REM LLEGADA AL NUEVO MUNDO
891 GOSUB 10000
892 GOSUB 10070
893 GOSUB 10300
894 GOSUB 10500
999 END
```

Aquí termina el programa principal. El resto del programa está escrito en forma de subrutinas a las que se llama desde esta sección principal

```
1000 PRINT CHR$(147):PRINT"ETAPA 1 — CONTRATAR LA TRIPULACION"
1010 PRINT  "-------------------- "
1012 PRINT
1015 GOSUB 9200
1020 PRINT:PRINT"CATEGORIAS DE TRIPULANTES DISPONIBLES:"
1025 GOSUB 9200
1030 PRINT
1040 PRINT  "CATEGORIA  DESCRIPCION  SALARIO SEMANAL"
1050 PRINT  "---------  -----------  ---------------"
1060 PRINT "           1   MARINERO          10 PIEZAS DE ORO"
1070 PRINT "           2  MEDICO             25 PIEZAS DE ORO"
1080 PRINT "           3  MECANICO           15 PIEZAS DE ORO"
1090 PRINT "           4  OFICIAL            20 PIEZAS DE ORO"
1100 PRINT "           5  COCINERO           15 PIEZAS DE ORO"
1105 GOSUB 9200
1110 PRINT:PRINT:PRINT
1120 S$="ENTRE LA CATEGORIA DE TRIPULANTE
     REQUERIDA(1-5)*":GOSUB 9100
1122 S$="0 'F' PARA FINALIZAR LA CONTRATACION*":GOSUB
     9100:PRINT:INPUT P$
1125 CT=VAL(P$)
1128 IF LEFT$(P$,1)="F"THEN PRINT:PRINT"FIN DE LA CONTRATACION DE
     TRIPULACION.":GOSUB 9200:GOTO 1310
1130 IF CT>0 AND CT<6 THEN 1150
1139 PRINT:PRINT
1140 PRINT P$;:S$="  NO ES UNA CATEGORIA DE TRIPULANTE*":GOSUB
     9100
1142 GOSUB 9200
1145 S$="INTENTELO NUEVAMENTE, POR FAVOR"
1146 GOSUB 9100
1147 GOTO 1300
1150 PRINT:PRINT
1155 CN=CN+1: REM TRIPULACION CONTRATADA HASTA AHORA
1156 TS(CN,1)=CT:REM CATEGORIA DE TRIPULANTE
1157 TS(CN,2)=100:REM FORTALEZA INICIAL
1158 WT=WT+WG(T):REM SALARIOS TOTALES
1159 CC(CT)=CC(CT)+1:REM CONTADOR CATEGORIAS DE TRIPULACION
1160 S$="TRIPULACION HASTA EL MOMENTO:"
1170 FOR T=1 TO 5
1180 PRINT S$;CC(T);"  ";C$(T);
1185 IF CC(T)>1 OR CC(T)=0THEN PRINT "S":GOTO 1189
1186 PRINT"      "
1189 S$="              "
1190 NEXT
1195 PRINT:PRINT"FACTURA SALARIAL SEMANAL TOTAL  ";WT
1200 IF CN=CM-1THEN PRINT:S$="SOLO UN TRIPULANTE MAS*":GOSUB
     9100: GOTO 1295
1202 IF CN=CM THEN PRINT:S$="     EL BARCO YA ESTA
     COMPLETO!!":GOSUB 9100:GOTO 1310
1295 REM
1300 GOTO 1015
1310 PRINT:S$=K$:GOSUB 9100:PRINT: GOSUB 9200
1320 GET P$:IF P$="" THEN 1320
1999 RETURN
2005 PRINT CHR$(147): REM LIMPIAR PANTALLA
2010 S$="            ETAPA  2 — APROVISIONAMIENTO*"
2015 GOSUB 9100
2020 S$=             -------     -----------------*"
2025 GOSUB 9100
2030 GOSUB 9200:PRINT
2040 PRINT"HAS CONTRATADO UNA TRIPULACION COMPUESTA
     POR";CN;"MIEMBROS."
2045 GOSUB 9200:GOSUB 9200
2050 FOR T=1 TO 4
2055 PRINT
2060 PRINT"CADA MIEMBRO DE LA TRIPULACION NECESITARA  "
2070 PRINT"  AL MENOS  ";PN(T);"  ";U$(T);
2075 IF PN(T)=1 THEN PRINT"  ";:GOTO 2085
2080 PRINT"S";
2085 PRINT"  DE  ";P$(T)
2086 PRINT"A  ";PC(T);"  PIEZAS DE ORO POR  ";U$(T)
2090 PRINT"POR CADA SEMANA QUE DURE EL VIAJE."
2095 GOSUB 9200:PRINT:GOSUB 9200
2100 PRINT"CUANTOS  ";U$(T);"S DE  ";P$(T)
2110 S$"DESEAS COMPRAR*":GOSUB 9100
2120 PRINT
2130 INPUT P$
2140 PA(T)=VAL(P$):GOSUB 9200
2150 IF PA(T)>((CN*8*PN(T))-1)THEN 2260
2160 IF PA(T)=0 THEN PRINT"SI NO COMPRAS NADA DE":GOTO 2180
2170 PRINT"SI SOLO COMPRAS  ";PA(T);U$(T);
2175 IF PA(T)=1 THEN PRINT"  DE":GOTO 2180
2176 PRINT"S DE"
2180 PRINT P$(T);",":GOSUB 9200
2190 PRINT"ALGUIEN PODRIA PASAR  ";
2200 S$="HAMBRE"
2210 IF T=4 THEN S$="SED"
2220 PRINT S$;"!":GOSUB 9200
2230 S$="QUIERES VOLVER A PROBAR (S/N)":GOSUB 9100
2240 INPUT P$:P$=LEFT$(P$,1)
2242 IF P$<>"S" AND P$<>"N" THEN 2230
2245 IF P$="N" THEN 2400
2250 PA(T)=0:T=T-1:GOTO 2410
2260 IF PA(T)*PC(T)>MO THEN 2270
2265 GOTO 2400
2270 S$="NO TIENES DINERO SUFICIENTE PARA":GOSUB 9100
2280 PRINT PA(T)
2290 PRINT U$(T);"S DE  ";P$(T):GOSUB 9200
2300 S$="POR FAVOR VUELVE A PROBAR*":GOSUB
     9100:PA(T)=0:T=T-1:GOTO 2410
2400 MO=MO-(PA(T)*PC(T))
2410 PRINT:S$="PROVISIONES HASTA AHORA:*":GOSUB 9100
2412 GOSUB 9200
2415 FOR TT=1 TO 4
2420 PRINT PA(TT);U$(TT)
2430 IF PA(TT)=1 THEN PRINT"   DE  ";:GOTO 2440
2435 PRINT"S DE  ";
2440 PRINT P$(TT)
2450 GOSUB 9200
2460 NEXT
2480 PRINT"DINERO QUE TIENES AUN=  ";MO;"  PIEZAS DE ORO"
```

```
2485 GOSUB 9200:GOSUB 9200
2490 NEXT T
2500 GOSUB 9200:PRINT:S$="FIN DEL APROVISIONAMIENTO*":GOSUB
     9100: GOSUB 9200
2510 PRINT:S$=K$:GOSUB 9100:PRINT : GOSUB 9200
2520 GET P$:IF P$="" THEN 2520
2999 RETURN
3000 REM **** ETAPA 3 - OTRAS MERCANCIAS ****
3001 PRINT CHR$(147):REM ETAPA 3
3002 GOSUB 9200
3005 PRINT"          ETAPA 3 — OTRAS  MERCANCIAS"
3010 PRINT"          -------     -----  ----------"
3020 GOSUB 9200
3025 PRINT
3030 S$="HAY OTRAS COSAS QUE PODRIAN*":GOSUB 9100
3035 S$="SERLE UTILES PARA EL VIAJE, POR *":GOSUB 9100
3040 S$="EJEMPLO, MEDICINAS Y MERCANCIAS*":GOSUB 9100
3045 S$="PARA COMERCIAR.*":GOSUB 9100
3046 GOSUB 9200
3050 S$="PUEDES NECESITAR TAMBIEN ESCOPETAS.*":GOSUB 9100
3055 GOSUB 9200:GOSUB 9200
3060 FOR T=1 TO 6
3065 PRINT
3070 PRINT"UN  ";D$(T);
3075 S$="  CUESTA*": GOSUB 9100
3080 PRINT OC(T);
3081 PRINT"  PIEZA DE ORO";
3085 IF OC(T)=1 THEN PRINT"PIEZA DE ORO":GOTO 3090
3086 PRINT "PIEZAS DE ORO"
3090 GOSUB 9200
3095 S$="TE GUSTARIA COMPRAR (S/N)*":GOSUB 9100
3110 INPUT P$:P$=LEFT$(P$,1)
3115 IF P$<>"S" AND P$<>"N" THEN 3095
3120 IF P$="N" THEN 3175
3125 GOSUB 9200
3130 S$="CUANTO QUIERES*":GOSUB 9100
3135 INPUT P$
3140 TT=VAL(P$)
3145 IF OC(T)*TT>MO THEN 3150
3147 GOTO 3160
3150 S$="NO TIENES DINERO SUFICIENTE*":GOSUB 9100
3152 GOSUB 9200
3154 S$="POR FAVOR VUELVE A ENTRAR*":GOSUB 9100
3155 GOSUB 9200:GOTO 3130
3160 MO=MO-(OC(T)*TT)
3165 OA(T)=TT
3170 GOSUB 9200
3175 PRINT
3176 PRINT"DINERO SOBRANTE=  ";MO
3200 GOSUB 9200:NEXT T
3205 GOSUB 9200:PRINT:PRINT
3210 S$="FIN DE LA ETAPA 3*":GOSUB 9100
3220 GOSUB 9200:PRINT
3230 S$=K$:GOSUB 9100
3240 GET P$:IF P$=""THEN 3240
3999 RETURN
```

Desde el bucle principal del viaje se llama a las siguientes subrutinas para analizar el estado actual del barco y la tripulación y confeccionar un informe semanal para el jugador

```
4000 REM INFORME SOBRE EL ESTADO DE LA TRIPULACION
4010 PRINT CHR$(147)
4020 S$="  DIARIO DE NAVEGACION DEL CAPITAN *":GOSUB 9100
4025 S$="  ------------------------------- *":GOSUB 9100
4030 GOSUB 9200
4035 PRINT"AL EMPEZAR LA SEMANA";WK
4040 S$="EL ESTADO DE LA TRIPULACION ES:*":GOSUB 9100
4045 GOSUB 9200:PRINT
4055 PRINT
4060 FOR T=1 TO 16
4070 IF TS(T,1)=0 THEN 4100
4075 PRINT C$(TS(T,1));"      (";
4078 IF TS(T,2)=-999 THEN S$="MUERTO !!!!!)*":GOTO 4099
4080 IF TS(T,2)>75 THEN S$="MUY SANO*":GOTO 4099
4085 IF TS(T,2)>50 THEN S$="SANO)*":GOTO 4099
4095 IF TS(T,2)>25 THEN S$="ENFERMO !)*":GOTO 4099
4098 S$="MUY ENFERMO !! )*"
4099 GOSUB 9100:GOSUB 9200
4110 NEXT T
4115 GOSUB 9200:PRINT
4119 WW=0
4120 FOR T=1 TO 5
4130 WW=WW+(CC(T)*WG(T))
4135 NEXT
4140 S$="FACTURA SALARIAL PARA LA SEMANA*":GOSUB 9100
4145 PRINT WW;"PIEZAS DE ORO"
4150 GOSUB 9200
4155 WT=WT+WW
4160 S$="TOTAL SALARIOS DEL VIAJE HASTA AHORA:*":GOSUB 9100
4165 PRINT WT;"PIEZAS DE ORO"
4170 GOSUB 9200
4175 PRINT"DINERO RESTANTE =";MO;"PIEZAS DE ORO"
4180 PRINT:S$=K$:GOSUB 9100
4190 GET I$:IF I$=" "THEN 4190
4199 RETURN
4200 REM INFORME SOBRE PROVISIONES
4205 PRINT CHR$(147)
4206 PRINT"EMPEZAR LA SEMANA";WK:GOSUB 9200
4210 S$="TE QUEDAN LAS SIGUIENTES*":GOSUB 9100
4215 S$="PROVISIONES:*":GOSUB 9100
4220 PRINT:GOSUB 9200
4225 FOR T=1 TO 4
4226 IF PA(T)=0 OR PA(T)=-999 THEN 4240
4230 PRINT PA(T);U$=(T);"S DE ";P$(T)
4232 X=INT(PA(T)/(CN*PN(T)))
4235 PRINT"(SUFICIENTE PARA";INT(X);" SEMANAS)"
4239 GOSUB 9200
4240 NEXT
4290 PRINT:S$=K$:GOSUB 9100
4295 GET I$:IF I$=""THEN 4295
4299 RETURN
4300 REM INFORME OTRAS MERCANCIAS
4305 PRINT CHR$(147)
4306 PRINT"AL EMPEZAR LA SEMANA";WK:GOSUB 9200
4310 S$="TAMBIEN POSEES:*":GOSUB 9100
4320 PRINT:GOSUB 9200
4322 IF OA(1)=0 THEN 4332
4325 PRINT OA(1);:S$="FRASCOS DE MEDICINA*":GOSUB 9100
4330 GOSUB 9200
4322 IF OA(2)=0 THEN 4342
4335 PRINT OA(2);:S$="ARMAS*":GOSUB 9100
4340 GOSUB 9200
4342 IF OA(3)=0 THEN 4352
4345 PRINT OA(3);:S$="SACOS DE SAL*":GOSUB 9100
4350 GOSUB 9200
4352 IF OA(4)=0 THEN 4362
4355 PRINT OA(4);:S$="BALAS DE TELA*":GOSUB 9100
4360 GOSUB 9200
4362 IF OA(5)=0 THEN 4372
4365 PRINT OA(5);:S$="CUCHILLOS*":GOSUB 9100
4370 GOSUB 9200
4372 IF OA(6)=0 THEN 4380
4375 PRINT OA(6);:S$="JOYAS*":GOSUB 9100
4380 GOSUB 9200:PRINT
4382 PRINT"TE QUEDAN";MO;:S$=" PIEZAS DE ORO*":GOSUB 9100
4384 GOSUB 9200
4397 PRINT:S$=K$:GOSUB 9100
4398 GET I$:IF I$="" THEN 4398
4399 RETURN
5000 REM COMPROBAR FACTURA SALARIAL
5005 IF WT>MO THEN 5010
5008 GOTO 5099
5010 PRINT CHR$(147)
5020 PRINT:PRINT:PRINT
5025 S$="ENTRE LA TRIPULACION CORRE EL RUMOR*":GOSUB 9100
5030 S$="DE QUE NO TIENES SUFICIENTE*":GOSUB 9100
5035 S$="ORO PARA PAGARLES CUANDO TERMINE*":GOSUB 9100
5040 GOSUB 9200:PRINT
5050 S$="LOS ANIMOS SE ESTAN EXALTANDO !!*":GOSUB 9100
5055 GOSUB 9200:PRINT
5060 S$="ESPEREMOS QUE CONSIGAS*":GOSUB 9100
5065 S$="OBTENER ALGUN BENEFICIO!*":GOSUB 9100
5066 GOSUB 9200
5070 PRINT:S$=K$:GOSUB 9100
5080 GET I$:IF I$="" THEN 5080
5099 RETURN
5100 REM REPARTIR RACIONES
5103 PRINT CHR$(147)
5105 S$="       REPARTIENDO LAS RAZONES*"GOSUB 9100
5106 S$="       -----------------------*"GOSUB 9100
5107 GOSUB 9200:PRINT"SEMANA";WK:PRINT
5108 H$="N"
5110 FOR T=1 TO 4
5112 HR(T)=1
5115 IF PA(T)>0 THEN 5180
5120 PRINT"NO QUEDA ";P$(T);" !!!!":GOSUB 9200
5130 S$="LA TRIPULACION SE ESTA DEBILITANDO !!*":GOSUB 9100
5135 WF=10:GOSUB 9300
5139 GOTO 5290
5180 X=(PN(T)*CN)*(JL-WK+1)
5185 IF PA(T)<X THEN 5200
5190 GOTO 5270
5200 PRINT"QUEDA POCA ";P$(T)
5205 GOSUB 9200
5210 S$="QUIERES PONER A LA TRIPULACION A*":GOSUB 9100
5215 PRINT"MEDIA RACION DE ";P$(T)
5220 INPUT I$:I$=LEFT$(I$,1)
5221 IF I$<> "S" AND I$<>"N" THEN 5220:REM ERROR EN ENTRADA
5225 IF I$="N" THEN 5270
5230 HR(T)=.5:H$="S"
5240 WF=5:GOSUB 9300
5250 S$="LA TRIPULACION SE ESTA DEBILITANDO!*":GOSUB 9100
5270 X=PN(T)*HR(T)*CN
5272 IF X>PA(T) THEN X=PA(T)
5275 PA(T)=PA(T)-X
5280 IF PA(T)=0 THEN PA(T)=-999
5285 PRINT X;U$(T);"S DE ";P$(T);" REPARTIDAS"
5290 PRINT:GOSUB 9200:NEXT
5295 PRINT:S$=K$:GOSUB 9100
5298 GET I$:IF I$="" THEN 5298
5299 RETURN
```

Ian Mckinnell



Puertos y canales en el Spectrum

Continuando con nuestro análisis del sistema operativo del Spectrum, vamos a ver cómo el ordenador envía los datos a la pantalla y a la impresora ZX por medio de canales

En el Spectrum de Sinclair el medio habitual de entrada de datos es el teclado, y las salidas habituales son una pantalla de televisor o una impresora ZX. Cada uno de estos instrumentos de hardware se denomina *canal* en el sistema del Spectrum. Así, la pantalla es llamada *canal de salida*. Los datos que fluyen desde o hacia el ordenador en forma de caracteres que van a la pantalla o vienen del teclado, son llamados una *corriente (stream)*. Una corriente o flujo de datos puede ser dirigida a través de los diferentes canales, en el supuesto de que el hardware de canales sea capaz de manejar la corriente de una manera correcta.

En el próximo capítulo hablaremos con más detalle de los canales y las corrientes. Mientras tanto, empezaremos por examinar el sistema de E/S, deteniéndonos en la rutina de "salida de un carácter", que se halla en la dirección &0010. El carácter enviado a esta rutina es lanzado a la pantalla o a la impresora, según el canal previamente seleccionado. El dibujo ilustra los diferentes canales de un Spectrum no ampliado y los números de las corrientes asociadas a ellos. Los canales son conocidos con una letra y las corrientes con un número. En el Spectrum no ampliado sólo actúan las corrientes 0, 1, 2 y 3. El OS dispone las corrientes y los canales conforme al cuadro que vemos en el margen.

Así, cuando deseamos dar salida a un carácter por un determinado dispositivo habremos de decirle al Spectrum en qué corriente lo deseamos. Para escribir en pantalla, tomaremos la corriente 2, ya que es la que se asocia con el canal S, para lo cual emplearemos una rutina ROM en la dirección &1601 que dirá al OS la corriente que hemos seleccionado. El número de la corriente se coloca en el registro A, antes de que sea llamada la rutina, y ésta abre posteriormente el canal hardware asociado actualmente con dicho número. Por ejemplo, para abrir el canal S para una salida, haremos

```
LD      A,2
CALL    &$1601
```

Una vez abierto el canal, el envío por él de un carácter se hace simplemente colocando el código del carácter en el registro A y ejecutando después una instrucción RST para llamar a la rutina con dirección &0010.

Corriente	Canal
0	K
1	K
2	S
3	P



Cruzar el canal

S: **Parte superior de la pantalla**
K: **Empleado en la salida para la parte inferior de la pantalla (el BASIC lo emplea para sus mensajes de error) y en la entrada desde el teclado**
P: **Impresora**

Con el Spectrum, los datos pueden transferirse desde los periféricos o hacia ellos en "corrientes" dirigidas por medio de diferentes canales asociados a ellas. Tras el encendido, los canales K, S y P se asocian a las corrientes 0 (y 1), 2 y 3 respectivamente. Así, PRINT #2 (es decir, la corriente 2: pantalla) equivale a la instrucción PRINT normal. Ejemplo: PRINT #2; "Esta nota se escribirá en la pantalla"

Esto se parece en algo a la llamada OSWRCH del BBC Micro, salvo que la llamada se hace directamente a la dirección de la ROM y no a través de un vector. El canal S opera sobre el área de la pantalla que es accesible con el PRINT normal del BASIC. Sin embargo también es posible acceder a las dos líneas inferiores que el intérprete del BASIC utiliza habitualmente para dar sus mensajes de error. Como se puede ver en el dibujo, estas líneas forman parte del canal K.

Para enviar un carácter a estas líneas, basta con que abramos el canal K, así:

```
LD    A,0
CALL  &1601
```

Igualmente, el canal de la impresora puede abrirse cargando el registro A con un 3.

No sólo los caracteres habituales pueden enviarse a través de cualquier canal, sino también los caracteres de control. El efecto resultante de estos últimos caracteres dependerá del canal empleado, pero esto significa que podemos obtener las equivalencias en código máquina de PRINT AT, PRINT INK, PRINT PAPER, etc. El cuadro que sigue muestra algunos códigos de control útiles y lo que realizan al ser pasados por el canal S o el K. Es claro que varios de ellos no tendrán efecto alguno si se envían a la impresora ZX (por medio del canal P).

Cod.	Parámetros	Efecto
8	—	Cursor en espacio a la izquierda
9	—	Cursor en espacio a la derecha
10	—	Cursor una línea hacia abajo
11	—	Cursor una línea hacia arriba
12	—	Delete (borrar)
13	—	ENTER
16	n	INK n (necesita un byte más)
17	n	PAPER n (neces. un byte más)
18	n	FLASH n (necesita un byte más)
19	n	BRIGHF n (neces. un byte más)
20	n	INVERSE n (nec. un byte más)
21	n	OVER n (necesita un byte más)
22	nn	AT y,x (necesita dos bytes más) para las coordenadas *x* e *y*)
23	n	TAB n (nec. un parámetro más)

Los bytes requeridos de más por algunos códigos de control son los parámetros que normalmente acompañan a éstos en una instrucción del BASIC. Así, por ejemplo, para ejecutar una orden PAPER 3, enviaríamos simplemente los bytes 17 y 3 al canal S. Este fragmento de programa muestra el equivalente en código máquina de PRINT AT 10,10;"A".

```
3E02      ld    a,2      ;abre canal S
CD0116    call  #1601    ;seleccionando corriente 2
3E16      ld    a,22     ;distintivo de AT
D7        rst   #10      ;rutina de sacar un carácter
3E0A      ld    a,10     ;ordenada y
D7        rst   #10
3E0A      ld    a,10     ;abscisa x
D7        rst   #10
3E41      ld    a,65     código ASCII de "A"
D7        rst   #10
C9        ret
```

No es difícil, que digamos. Una salvedad notable en esta lista de códigos de control la constituye el código para CLS que no está. Para borrar la pantalla debemos llamar a toda una rutina ROM de dirección &06DB. Es esencial que el canal S esté abierto antes de llamarla, y es también necesario que reabramos el canal S después de haberlo usado si deseamos que imprima cualquier otra cosa en la pantalla. Esta rutina borrará la pantalla:

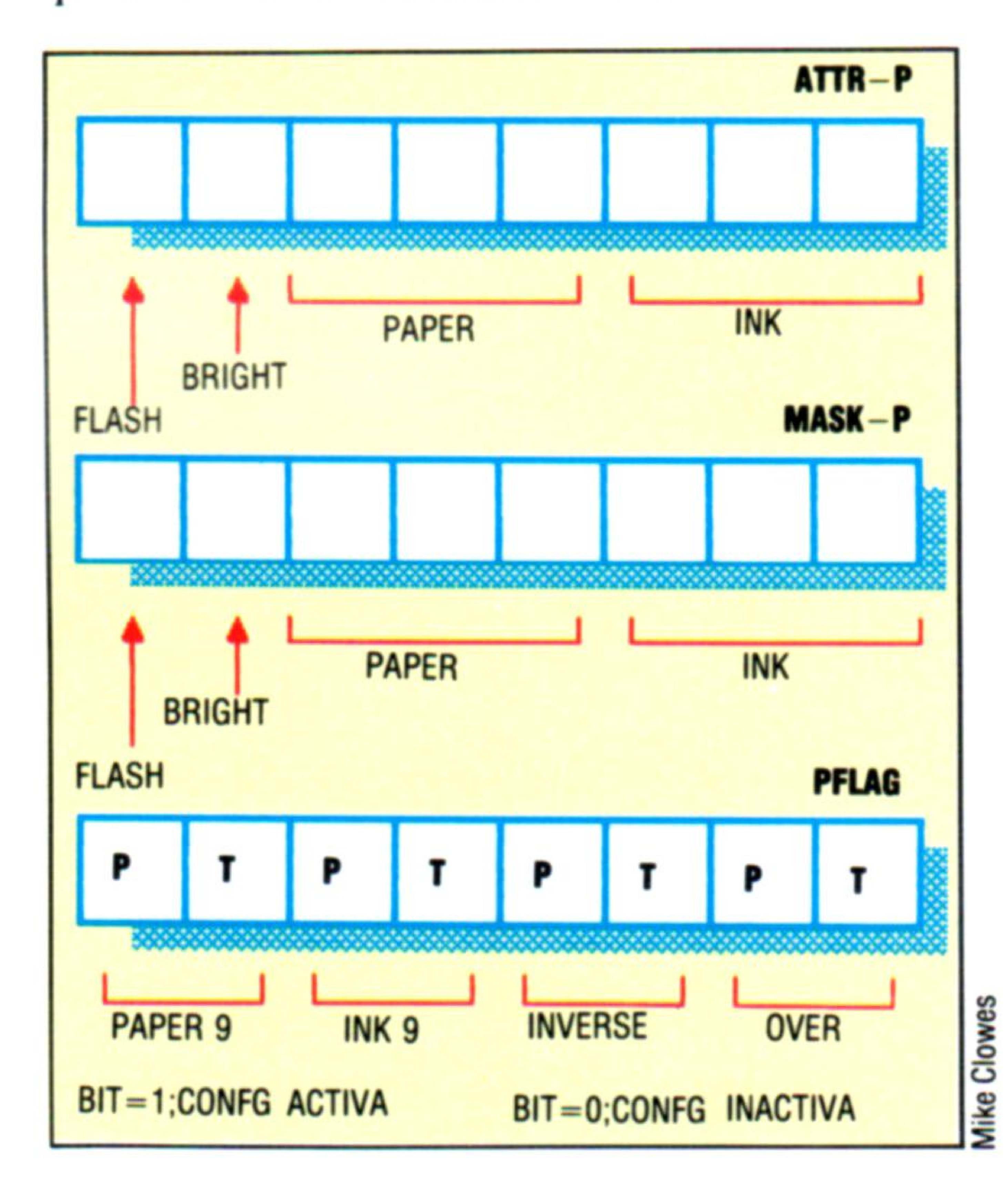
```
3E02      ld    a,2
CD0116    call  #1601    ;abre canal S
CDDB06    call  #06db    ;borra pantalla
3E02      ld    a,2
CD0116    call  #1601    ;reabre canal S
C9        ret
```

Una cualidad útil de la rutina "salida de un carácter" es que los números pasados a ella que representan instrucciones del BASIC son transmitidos por la rutina e impresos completamente. Así:

```
LD    A,249
RST   &10
```

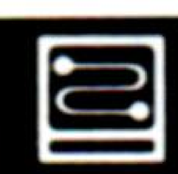
imprimirá, si se seleccionó el canal S, P o K, la instrucción RANDOMISE en la pantalla.

Las instrucciones para gráficos, como PAPER, INK y BRIGHT, realizadas a través de la rutina "salida de un carácter" son sólo operativas para la secuencia de salida de caracteres (se dice que son *ítems temporales del color*). La instrucción PAPER n, fuera de una sentencia PRINT, es *permanente* y opera hasta que se emite otra instrucción PAPER.



Las dos variables de sistema que nos interesan son ATTR–P y MASK–P. El esquema anterior muestra cómo estas variables controlan diferentes aspectos de la visualización. En ATTR–P los tres bits que controlan el color de PAPER y de INK de manera permanente se dan, con sus valores respectivos, en el cuadro de la página siguiente, encima de la primera columna de texto. Si el bit de FLASH o BRIGHT es uno, la configuración es operativa. La variable ATTR–P se halla en la dirección 23693.

MASK–P tiene de dirección 23694 y todo bit puesto a uno en este byte hace que los atributos en pantalla situados en la posición relevante de impresión no se alteren con el contenido de ATTR–P. Otra variable importante es la PFLAG que está en la dirección 23697. Damos también un cuadro sobre



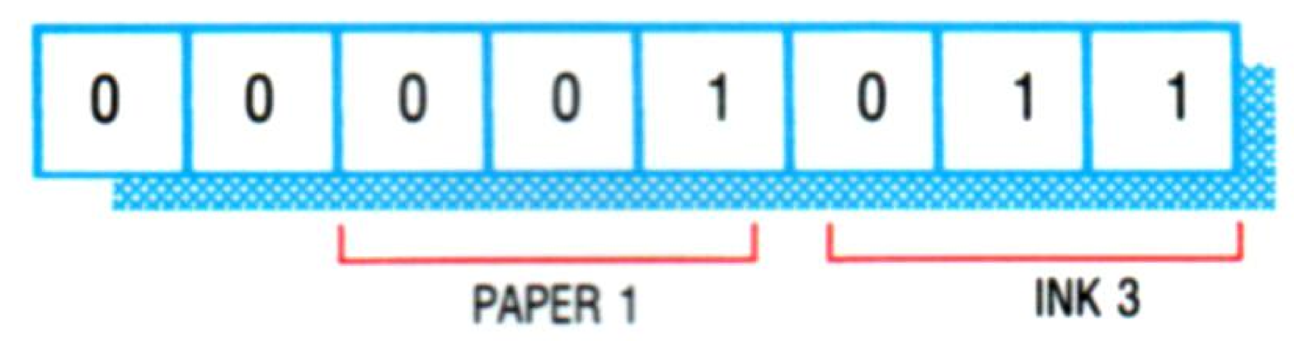
Bits	Color
000	Negro
001	Azul
010	Rojo
011	Magenta
100	Verde
101	Cyan
110	Amarillo
111	Blanco

esta variable. El OS la usa para indicar PAPER 9, INK 9, INVERSE y OVER.

Hay además dos variables en las direcciones 23695 y 23696 y son ATTR−T y MASK−T. Se disponen de modo semejante a las variables ATTR−P y MASK−P, pero controlan los colores temporales (los usados en las sentencias PRINT y en los establecidos enviando códigos de control por medio de la rutina RTS &10).

Para establecer colores mediante ATTR−P y MASK−P basta con manipular el contenido de las variables de sistema, alterando sólo los bits que nos interesen.

Esto se puede realizar fácilmente en el Z80 mediante las operaciones lógicas AND y OR. Así, para ejecutar una instrucción permanente PAPER 1:INK3, estableceremos la variable ATTR−P de la siguiente manera:



mediante este código:

```
LD     A,11
LD     (23693),A
RET
```

Naturalmente, como ocurre con las instrucciones permanentes PAPER e INK del BASIC, los nuevos colores no afectarán a lo que ha sido impreso antes y la pantalla no irá al nuevo color PAPER mientras no se ejecute una instrucción CLS o equivalente.

Las rutinas de gráficos en el Spectrum forman parte del programa intérprete del BASIC. Veamos ahora las operaciones PLOT y DRAW. Al emplear estas rutinas no es difícil realizar cambios de colores alterando las variables ATTR−P y MASK−P, según acabamos de describir.

Las rutinas en sí son fáciles de emplear. La primera, PLOT, se llama en la dirección &22E5. Sus coordenadas le son pasadas mediante el par de registros BC (B para la ordenada *y*, C para la abscisa *x*). Así, para ejecutar PLOT 100,100 desde un programa en código máquina simplemente ejecutaremos lo siguiente:

```
LD      B,100   ;ordenada y
LD      C,100   ;abscisa x
CALL    &22E5   ;lo ejecuta
RET
```

Los cambios de color se realizan con facilidad. La siguiente rutina traza un punto rojo en la pantalla y después restaura ATTR−P a su estado previo antes de volver al BASIC.

```
3A8D5C   10    ld    a,(23693)   ;toma ATTR-P del reg A
F5       20    push  af          ;y lo lleva a la pila
E6F8     30    and   248         ;pone los 3 bits inf a 0
F602     40    or    2           ;pone bit 1 para tinta roja
0664     50    ld    b,100       ;ordenada y
0E64     60    ld    c,100       ;abscisa x
CDE522   70    call  #22e5       ;llama a PLOT
F1       80    pop   af          ;restaura el contenido
328D5C   90    ld    (23693),a   ;original de ATTR-P
C9       100   ret
```

Hasta ahora nos hemos centrado en el canal S. ¿Qué decir del canal K y sus facilidades de salida? Si se quiere, es posible escribir en la parte inferior de la pantalla, pero si el OS o el intérprete genera un mensaje, éste se sobreimprime. Si empleamos este canal para entradas, no se necesita ni siquiera llamar a la rutina de ROM. El teclado es inspeccionado cada 20 microsegundos y se afectan algunas variables de sistema según se haya pulsado o no alguna tecla. Dos ejemplos de empleos son LAST−K (con dirección, 23560), que retiene el código del carácter correspondiente a la tecla últimamente pulsada, y una variable de sistema con dirección 23556, que retiene el valor 255 si en ese momento *no* se ha pulsado tecla alguna. Se puede usar la rutina para que espere hasta que se pulse una tecla y después lleve el código del carácter al registro A. Consiste sencillamente en comprobar el contenido de la dirección 23556 y ver cuándo no es 255. El valor contenido en LAST−K será en ese momento el de la tecla recién pulsada.

```
         10 ;rutina OBTENCION CARAC
3A045C   20 key:   ld   a,(23556)   ;comprueba si se pulsó
FEFF     30        cp   255         ;alguna tecla
28F9     40        jr   z,key       ;siguen comprobando
3A085C   50        ld   a,(23560)   ;lleva LAST-K al reg A
C9       60        ret
```

Antes de abandonar el teclado, repasemos un par de variables de sistema de bastante utilidad.

Variable	Dirección	Descripción
REPDEL	23561	Retardo de un 50avo de segundo antes de que el teclado comience repetición. Se puede alterar este valor y el retardo antes de que el teclado inicie la repetición
REPPER	23562	Retardo entre la segunda repetición del teclado y las siguientes. También se puede alterar (POKE)
PIP	23609	Duración del sonido que acompaña la tecla pulsada. Un valor mayor genera un *pitido*

Otro canal del Spectrum no ampliado que nos será útil es el canal P, empleado por la impresora ZX. Tiene habitualmente 3 corrientes asociadas. De este modo:

```
LD    A,3
CALL  &1601
```

abrirá el canal P y si se conecta la impresora ZX se imprimirán los caracteres que sigan. Está claro que el BASIC hace un uso muy amplio de los diferentes canales: PRINT y LIST utilizan el canal S, LPRINT y LLIST el canal P e INPUT utiliza el canal K. Podemos añadir nuestros propios canales a estos tres que acabamos de examinar, de manera que podamos acceder con facilidad a algún dispositivo adicional como el microdrive, otras impresoras u otras configuraciones hardware.

En el próximo capítulo examinaremos este punto con mayor atención.

Tacto vital

El control del cursor mediante la selección de iconos le confiere a "Shadowfire" una excitante dimensión

Shadowfire: Para Commodore 64 y Sinclair Spectrum (ambas versiones en la misma cassette)
Editado por: Beyond Software, Competition House, Farndon Road, Market Harborough, LE16 9NR, Gran Bretaña
Autores: S. Cain, D. Colcough, K. Davies, G. Everett, J. Gibson, F. Gray, J. Heap, A. Noble y C. Parrott
Palanca de mando: Opcional
Formato: Cassette

Hasta el observador más indiferente se habrá percatado ya de que los juegos por ordenador se están volviendo cada vez más sofisticados. En lugar de reproducir servilmente juegos de estilo recreativo o de limitarse a juegos de aventuras basados sólo en texto tomados del formato de *Calabozos y dragones*, los diseñadores de juegos para ordenadores personales están desarrollando su propio estilo. Estos juegos combinan muchas de las características de los formatos recreativo y de estrategia para producir una diversión que dura bastante más de los cinco minutos más o menos que dura la acción recreativa, y que es más exigente incluso que los intelectualmente complejos enigmas de los juegos de aventuras.

El argumento de *Shadowfire* básicamente es el siguiente: el general Zoff, un desertor, ha capturado al embajador Kryxix, quien posee los planos para un nuevo tipo de nave espacial denominada *Shadowfire*. El jugador ha de rescatar el embajador antes de que éste se vea forzado a revelar los planes, y dispone de 100 minutos para cumplir su misión. Para ayudarlo a realizar el rescate cuenta con seis personajes, cada uno de los cuales posee diferentes puntos fuertes y débiles.

Para introducirse a bordo de la nave espacial de Zoff, su equipo debe ser "teletransportado". El único miembro capaz de organizar esto es el zángano Manto, de modo que primero debe ser enviado él para que tienda el haz teletransportador que seguirán los demás.

Sin embargo, antes de que usted los despache, es aconsejable proveer a cada personaje de las armas disponibles, en función de sus cualidades y su "talón de Aquiles".

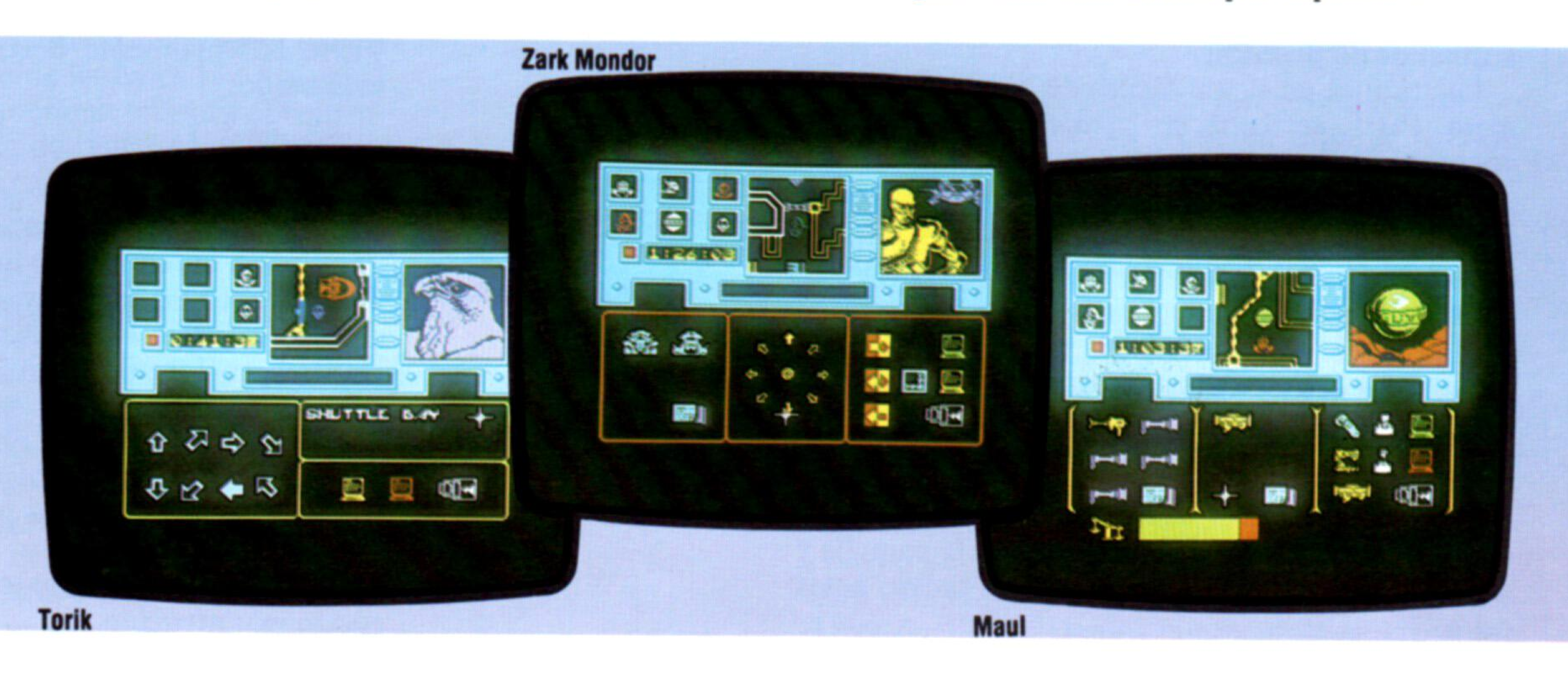
Shadowfire utiliza un sistema exclusivo para desplazar los personajes, permitiéndoles recoger objetos y luchar. A diferencia de otros programas de aventuras, que exigen que el usuario digite instrucciones tales como "avanzar norte" o "recoger láser", este juego permite llevar a cabo todas las acciones mediante iconos y un cursor móvil: algo así como el sistema operativo empleado en el Apple Macintosh. Por ejemplo, para proporcionar granadas de mano al líder del equipo, Zark, usted debe seleccionar el icono de éste. Tras elegirlo, la pantalla pasa a una visualización gráfica de su fuerza, su estamina y otros atributos. En el lado derecho de la pantalla hay tres iconos "monitores" que representan el movimiento, la modalidad de batalla y la pantalla de objetos.

Seleccionando la pantalla de objetos con el cursor (que se puede mover desde el teclado, mediante la palanca de mando o un lápiz óptico), ésta cambia otra vez, pasando a visualizar los objetos que se hallan en las inmediaciones del personaje, así como varios iconos de "actividades". Escogiendo el icono "recoger" y desplazando después el cursor hasta el icono de la granada, se equipará a Zark con granadas.

Una vez armado y "teletransportado" hasta la nave espacial, el equipo comenzará a buscar al general Zoff y al embajador Kryxix. La nave espacial se compone de numerosas habitaciones y pasillos, algunos de los cuales contienen armas o llaves que le permitirán abrir puertas cerradas, mientras que otros ocultan guardias enemigos, a los que habrá de destruir antes de poder seguir adelante. Si usted carece de la llave apropiada, puede requerir los servicios de Sevrina, quien se especializa en hacer saltar cerraduras.

Para emplear de la mejor forma posible los 100 minutos de que dispone, debe trazar una estrategia. Algunos personajes se desenvuelven mejor que otros en ciertas situaciones y, por lo tanto, es recomendable tener a la persona correcta en el lugar adecuado y en el momento oportuno.

Lo que hace de *Shadowfire* un juego tan interesante es el control por cursor. Con el proceso de selección de iconos, el jugador puede reaccionar ante las situaciones con mayor rapidez que si hubiera de digitar cada instrucción por separado.



Los protagonistas de la acción
Vemos aquí, en situaciones diferentes, a tres de los personajes que el jugador tiene bajo su control en *Shadowfire*. A la izquierda, Torik se halla en modalidad de movimiento, con el sombreado indicando las posibles direcciones que puede tomar. En el centro, Zark Mondor es atacado y se halla en modalidad de ataque. Nuevamente, se destacan las posibles direcciones que puede tomar y, a la izquierda de la pantalla, aparecen sus adversarios en forma de iconos. Maul, un droide de batalla, aparece con una pantalla de objetos. La sección del centro de la parte inferior de la visualización muestra los objetos que lleva actualmente consigo, y a la izquierda aparecen los objetos que se hallan cerca de él y que puede recoger. Las instrucciones se le envían al ordenador desplazando un cursor (una cruz blanca) hasta el icono seleccionado y pulsando el botón de disparo de la palanca de mando

mi computer

Editorial Delta, S.A.



LIBROS PARA TU MICROORDENADOR

18 JUEGOS DINÁMICOS PARA TU ZX SPECTRUM
por P. Monsaut, P.V.P. 650 Ptas.

En este libro se presenta una colección de 18 programas de juegos variados que combinan todas las posibilidades de su ordenador, color, gráficos, movimiento, etc. Además no sólo se limita a presentar juegos sino que aprovecha para mostrar algunos trucos y técnicas de programación.

18 JUEGOS DINÁMICOS PARA TU DRAGON 32
por P. Monsaut, P.V.P. 650 Ptas.

En este libro se presenta una colección de 18 programas de juegos variados que combinan todas las posibilidades de su ordenador, sonido, color, gráficos, movimiento, etc. Además no sólo se limita a presentar juegos sino que aprovecha para mostrar algunos trucos y técnicas de programación.

18 JUEGOS DINÁMICOS PARA TU COMMODORE 64
por P. Montsaut, P.V.P. 650 Ptas.

En este libro se presenta una colección de 18 programas de juegos variados que combinan todas las posibilidades de su ordenador, sonido, color, gráficos, movimiento, etc. Además no sólo se limita a presentar juegos sino que aprovecha para mostrar algunos trucos y técnicas de programación.

OTROS TITULOS

ZX SPECTRUM - QUÉ ES, PARA QUÉ SIRVE Y CÓMO SE USA
por Tim Langdell, P.V.P. 1.100 Ptas.

ZX SPECTRUM - APLICACIONES PRÁCTICAS PARA LA CASA Y LOS PEQUEÑOS NEGOCIOS
por Chris Callender, P.V.P. 870 Ptas.

PROFUNDIZANDO EN EL ZX SPECTRUM
por Dilwyn Jones, P.V.P. 1.300 Ptas.

CÓMO CREAR TUS JUEGOS SPECTRUM
por R. Rovira, P.V.P 750 Ptas.

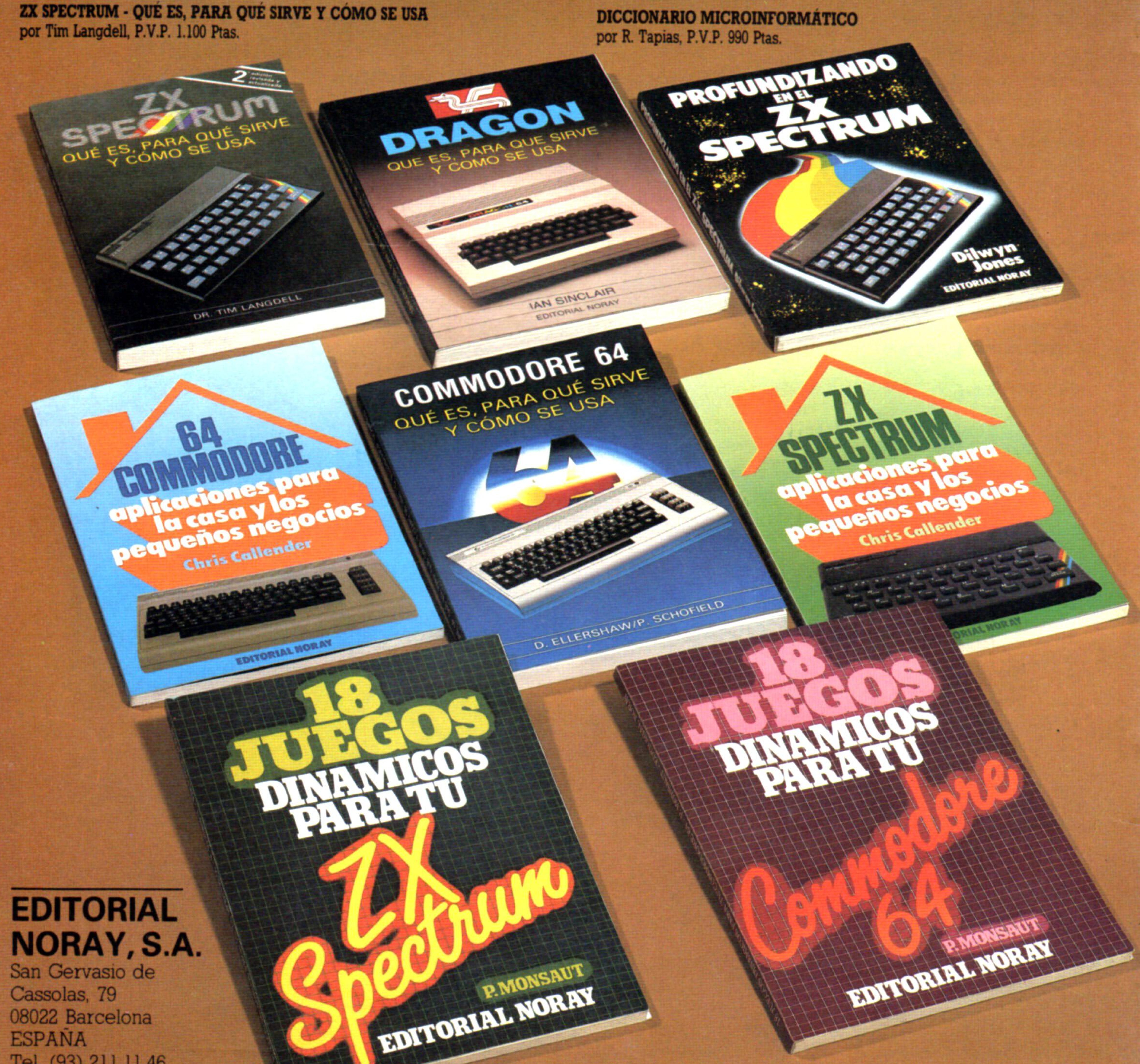
DRAGON - QUÉ ES, PARA QUÉ SIRVE, CÓMO SE USA
por Ian Sinclair, P.V.P. 1.300 Ptas.

COMMODORE 64 - QUÉ ES, PARA QUÉ SIRVE Y CÓMO SE USA
por D. Ellershaw y P. Schofield, P.V.P. 950 Ptas.

COMMODORE 64, APLICACIONES PRÁCTICAS PARA LA CASA Y LOS PEQUEÑOS NEGOCIOS
por Chris Callender, P.V.P. 830 Ptas.

INTRODUCCIÓN AL MSX
por Vanryb y Politis, P.V.P. 1.100 Ptas.

DICCIONARIO MICROINFORMÁTICO
por R. Tapias, P.V.P. 990 Ptas.



EDITORIAL NORAY, S.A.
San Gervasio de
Cassolas, 79
08022 Barcelona
ESPAÑA
Tel. (93) 211 11 46